1979-03

ATHENA : a system to interactively analyze large scale optimization models.

Galatas, Panagiotis Ioannou

Monterey, California : Naval Postgraduate School

ATHENA: A SYSTEM TO INTERACTIVELY ANALYZE LARGE SCALE OPTIMIZATION MODELS

Panagiotis Ioannou Galatas

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ATHENA: A SYSTEM TO INTERACTIVELY ANALYZE LARGE SCALE OPTIMIZATION MODELS

by

Panagiotis Ioannou Galatas

March 1979

Thesis Advisor: G. Bradley

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ATHENA: A System to Interactively Analyze Large Scale Optimization Models		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 1979 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Panagiotis Ioannou Galatas		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March 1979 13. NUMBER OF PAGES 101
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Optimization Model

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The analysis of the solutions of large scale optimization models is very difficult without computer aids because typical outputs may exceed 40,000 lines. A computer system to allow efficient storage and interactive analysis of the solution file was designed and implemented in FORTRAN. This

(20. ABSTRACT Continued)

system, called ATHENA, has been initially designed for problems with up to 30,000 rows and columns; tests show the system has fast response time for these large problems. A description of the system, its data structures and commands as well as a user's manual is included.

Approved for public release; distribution unlimited.

ATHENA: A System to Interactively Analyze Large Scale Optimization Models

by

Panagiotis Ioannou Galatas
Captain, Hellenic Army
B.S., Military Academy of Greece, 1965
M.S., Highest School of Economic Sciences of Athens, 1976

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN OPERATIONS RESEARCH
MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

March 1979

ABSTRACT

The analysis of the solutions of large scale optimization models is very difficult without computer aids because typical outputs may exceed 40,000 lines. A computer system to allow efficient storage and interactive analysis of the solution file was designed and implemented in FORTRAN. This system, called ATHENA, has been initially designed for problems with up to 30,000 rows and columns; tests show the system has fast response time for these large problems. A description of the system, its data structures and commands as well as a user's manual is included.

TABLE OF CONTENTS

ACKNOWLEDGMENT

The author would like to acknowledge the contributions of Professor Gordon Bradley whose mathematical programming expertise was of invaluable assistance in analyzing this problem, Professor Gerald Brown for his valuable guidance and the author's wife Lina Galatas who provided the encouragement and secretarial skills necessary for successful completion. The system name ATHENA has been chosen for several reasons. First, ATHENA is the goddess of wisdom from ancient Greece. Second, it is the Greek name of the capital of my country. Finally, ATHENA is my mother's name.

I. INTRODUCTION

The generation, solution and analysis of large scale mathematical programming models presents significant problems in data handling. For a typical large scale model the output from the computer may exceed 40,000 lines and it is very difficult and time consuming for the user to go through that much paper to extract the information he needs. Space is also needed to store all these outputs and it is difficult to provide routine access to old solution files.

Over the past several years, faculty and students at the Naval Postgraduate School and the University of California at Los Angeles have been cooperatively developing theory and algorithms to solve large scale linear, nonlinear and integer optimization problems. This research has been furthered by the development of several software systems to solve large scale optimization problems. ATHENA is part of that development and was built to satisfy a pressing need to be able to quickly and easily analyze solutions to these large problems.

The research in large scale optimization at the Naval Postgraduate School has concentrated on providing economic solutions to current Department of Defense problems. One such project that was going on concurrent with the development of ATHENA was a large, medium range capital budgeting problem that required the solution of a mixed integer programming

problem with 11,687 constraints and variables [6]. ATHENA was used successfully with this project, and performance of ATHENA on this problem is reported below.

System ATENA has been developed to handle output from these large optimization problems by enabling the user to get the information he needs interactively through a computer terminal and by economically storing the large files in packed form in low cost media. The features of the system may be summarized as follows:

1. Quick and accurate answers to simple questions that are tedious and error prone to address manually, e.g.,

How many of a set of variables are = 1.0?

or How many are greater than 0?

or What variables are in a specified range?

or What constraints are satisfied exactly? Etc.

2. In large scale mathematical programming models the names of rows and columns are constructed systematically so that groups of variables with relationships in the real world have similar names. Using the system one can have automotic, easy and accurate answers for many interesting properties of these groups. (For instance, the average value of all the variables whose names begin with X, etc.)

3. ATHENA can also be used as a basis for a simple report writer.

4. The system provides very compact computer storage: the original solution file is typically packed into 1/10 of its normal volume. For example, a solution file from the

IBM MPS/360 package of a linear programming model with 12,000 rows and columns occupies 1.5 magnetic tapes 2400 feet long at 800 BPI in original unblocked form, but in packed form the solution occupies approximately 31 feet of magnetic tape.

5. There is systematic and economic file organization with easy and accurate access.

6. A file structure for multiple runs of the problem for comparisons is available.

7. The system requires modest resources (core, time) in a time sharing environment for a large amount of data.

8. The system has portability and allows easy change or expansion since the whole system has been developed in FORTRAN.

ATHENA was inspired by a similar system developed for the Department of Energy by O'Neil and Sanders [5] called PERUSE. PERUSE was developed to aid in the analysis of large linear programming energy models. A study of the needs of Naval Postgraduate School students and faculty showed that additional capacities beyond those in PERUSE were necessary to support current and future research in large scale optimization. In addition to the standard MPS output, it was determined that ATHENA should support the experimental optimization system XS [3]; the output of this system contains in addition to standard MPS output, upper and lower penalties that implement the 'elastic formulation' of linear models that is unique to XS. ATHENA also had to support the

use of a preprocessor PREP [2] that reformulates the original optimization problem to an equivalent reduced problem with fewer rows and/or columns. A study of past and current modeling efforts at the Naval Postgraduate School identified additional commands that would help in the analysis of large models.

ATHENA was designed to be as much as possible a direct extension of PERUSE. Almost all the commands and options of PERUSE have been included with the identical names and syntax whenever possible. A summary of the extensions is listed under the section LIMITATIONS - EXTENSIONS, of the user's manual.

II. SOFTWARE DESCRIPTION

A. GENERAL

The whole system consists of 3 basic subsystems (see Figure 1).

1. PARSER - Code Generator

This subsystem accepts as input a Query, parses it examining the syntax according to the productions of the Query Language and generates the corresponding internal code or gives information for syntax errors. The internal codes for each Query are shown in the program list.

2. Input

This subsystem accepts as input either (1) an unpacked solution file in 'standard' format which it packs and saves for future reference, or (2) an L.P. solution file in packed form from a previous session.

3. Interpreter

This subsystem, using the code generated from the PARSER, searches the packed solution file and prints out the information requested.

B. LANGUAGE

1. Host Language

The system has been developed in a portable subset of FORTRAN IV. FORTRAN was chosen for the following reasons:

a. FORTRAN is a general language available at almost any computer installation, so the system can be used with

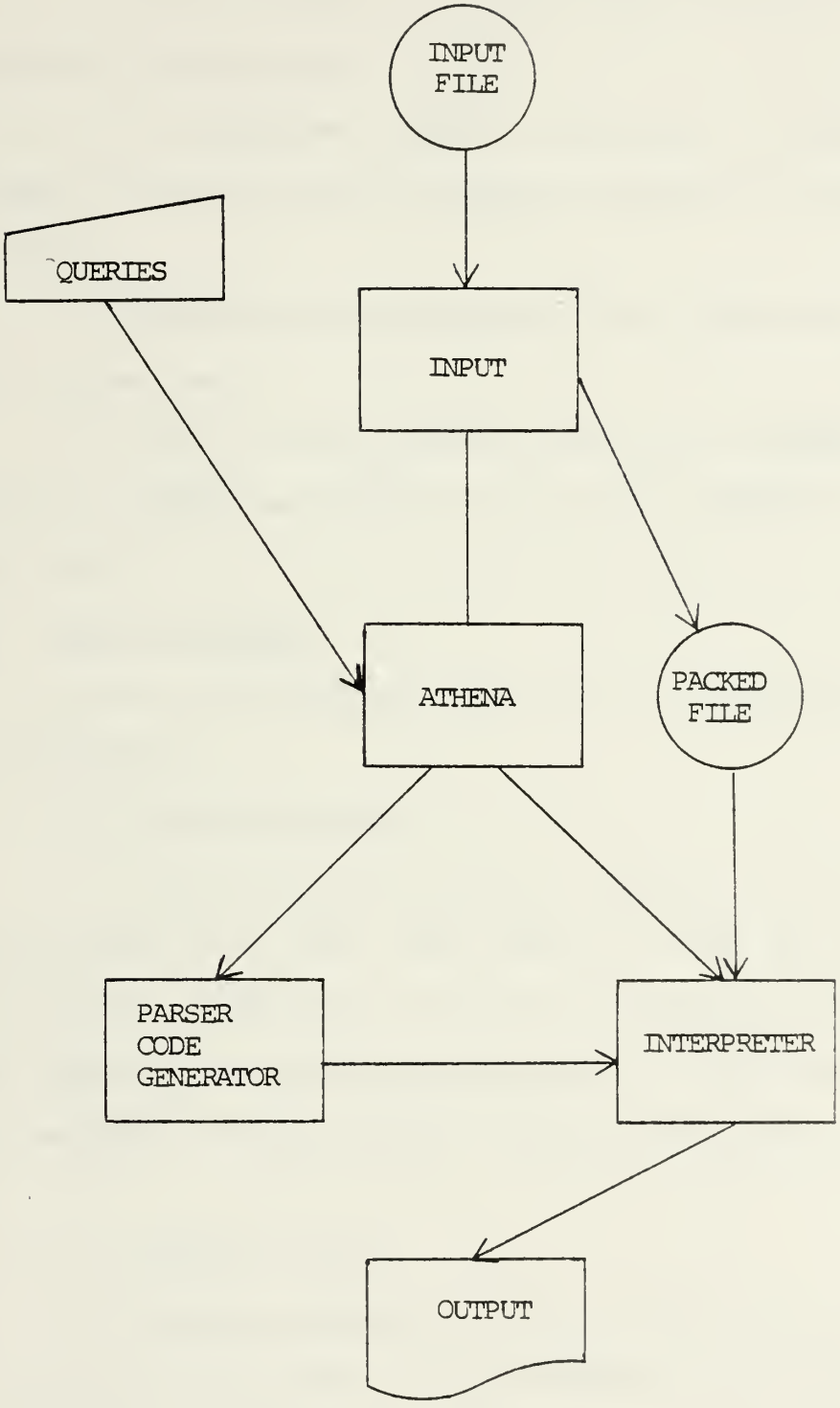


FIGURE 1. System ATHENA

any contemporary hardware. Non-IBM systems may require some program modifications.

b. Since FORTRAN is a high level language, the development time was low.

c. The response time for each Query is acceptable and there is little need for faster responses or enhanced efficiency.

d. Extensions and changes of the system can be easily implemented.

e. There is good system support for FORTRAN. In particular, ATHENA was developed with the FORTRAN G compiler of IBM 360/67.

2. Query Language

The set of acceptable Queries is divided into two main categories:

a. Control Queries

Control Queries provide commands to the system to perform specific tasks, but usually do not use the solution file. Examples of Control Queries are those that accept comments for self documentation of the output, print headings for the output, terminate the use of system, etc.

b. Command Queries

Command Queries use the solution file to extract the information asked for. Each Command Query consists of various fields separated by at least one blank. Some of the fields are optional, while others are required. An internal code number is generated by parsing the Command Query for

each field depending upon the analysis of that field and the previous fields in the Command Query.

The code generated by the PARSER is executed by the INTERPRETER which consists of a set of progams (subroutines) activated by the code numbers, to get the required information from the file.

C. DATA STRUCTURES

1. Preview

There are some observations about the solution file of a linear program, especially of a large scale one, that lead to the use of a special data structure for storing a solution file in less memory space than it would otherwise require.

For each row or column the following information is usually included in the solution file.

NUMBER of row or column.

NAME, usually 6-8 alphanumeric characters.

STATUS, usually 2 characters, e.g., BS for BASIC, LL for LOWER LIMIT, etc.

ACTIVITY LEVEL, for each row or column.

SLACK ACTIVITY for rows or INPUT COST for columns.

LOWER LIMIT

UPPER LIMIT

DUAL ACTIVITY for rows or REDUCED COST for columns.

UPPER PENALTY and

LOWER PENALTY for the elastic linear programming system, XS [3].

There is redundant information in each record. Some of these redundancies are the following:

The explicit number for each row or column may be represented implicitly by the ordinal position of the row or column in the file. Rows usually precede columns in solution files.

A large number of the ACTIVITY LEVEL values will be zero. The same is true for the SLACK ACTIVITY, LOWER LIMIT, DUAL ACTIVITY and PENALTY values.

In many cases there will not be LOWER or UPPER LIMITS or PENALTIES.

PENALTIES in some cases may be infinite.

When the status of a row or column is 'fixed', then ACTIVITY LEVEL, LOWER and UPPER LIMITS are all the same number.

Each row or column can be in only one of its possible states.

Moreover, analysts who have experience with large scale Linear Programming have observed that most of the numbers of the solution file are the same. For example, most of the numbers for LIMITS are the same for a large number of rows or columns. For purposes of analysis, it is rarely necessary to have more than five decimal digits of precision for problem values. Indeed, some large problems cannot be solved with even this degree of significance. Accordingly, IBM single precision REAL*4 representation

is adequate for our purposes. Conversion to REAL*8 extended precision requires trivial program modifications.

Based on the above observations, two types of data structures for storing the solution file have been developed. The first one (SPARSE) exploits the redundant information into each individual record. The second (SUPERSPARSE) takes advantage of the last observation by storing each distinct number only once for the entire file. It is the responsibility of the user to select the data structure type that is appropriate for each solution file. SUPERSPARSE is probably superior for problems with less than half of all numbers possessing distinct real values. What follows is a detailed description of these two data structures.

2. SPARSE Data Structure

The entire solution file is stored in contiguous memory (8-bit bytes) as a one-dimension array called SOLFIL, in the following way:

a. The first 16 bytes (Four 4-byte words) are used to keep information for:

(1) The size of the file in 4-byte words.

(2) The type of data structure used to pack the file (SPARSE or SUPERSPARSE).

(3) The number of rows and columns of the file.

b. For each row or column, 12 sequential bytes are required, organized as follows (see Figure 2).

(1) The first 8 bytes hold the name of the row or column left justified, one character per byte.

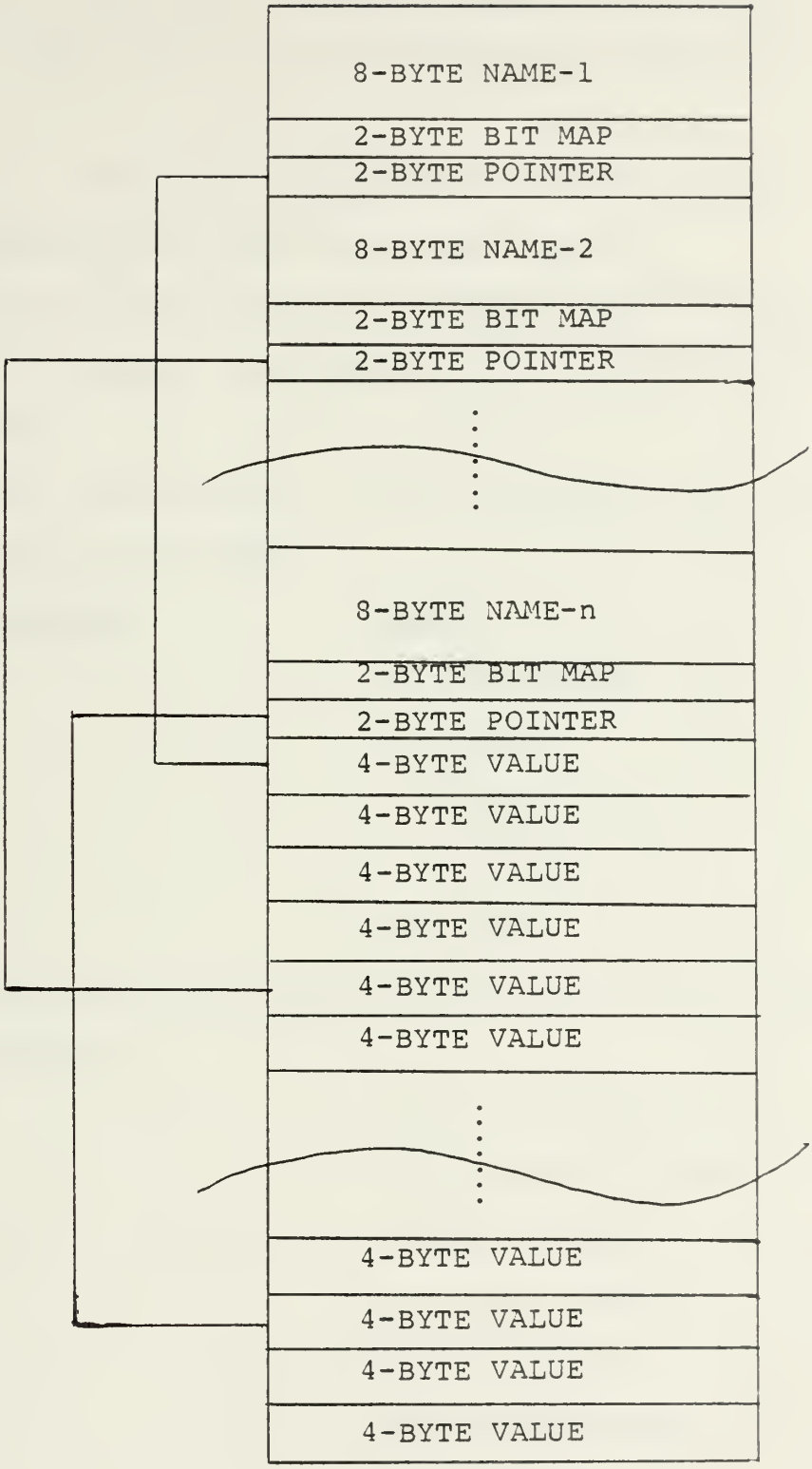


FIGURE 2. SPARSE Data Structure

(2) The next 2 bytes are used (as 16 bits) to represent various characteristics associated with that row or column.

(3) The last 2 bytes are used as a pointer to the first number stored from the current record.

c. The 16 bits from (2) above are organized in 4 groups of 4, 7, 4 and 1 bits, respectively, taken from higher to lower order.

The first group of 4 bits represents the status of the current row or column.

BIT PATTERN	STATUS
0 0 0 0	IN (INFEASIBLE)
0 0 0 1	BS (BASIC)
0 0 1 0	LL (LOWER LIMIT)
0 0 1 1	UL (UPPER LIMIT)
0 1 0 0	EQ (FIXED)

The following status indicators are reserved for use with the program PREP [2]

0 1 0 1	VC (VOID COLUMN)
0 1 1 0	SC (SINGLETON COLUMN)
0 1 1 1	FC (FIX COLUMN)
1 0 0 0	BC (BOUND CHANGED)
1 0 0 1	VR (VOID ROW)
1 0 1 0	SR (SINGLETON ROW)
1 0 1 1	RR (REDUNDANT ROW)
1 1 0 0	FR (ROW FIXES VAR. AT BOUND)
1 1 0 1	ER (DOUBLETON EQUATION)

1 1 1 0	TR (TIGHTEN RANGE)
1 1 1 1	PP Reserved for PREP[2]

The next group of 7 bits represents the characteristic of zero or nonzero values for the record.

BIT NO:	BIT VALUE:	CHARACTERISTIC
11	0	ACTIVITY LEVEL NONZERO
	1	ACTIVITY LEVEL ZERO
10	0	SLACK/COST NONZERO
	1	SLACK/COST ZERO
0	0	LOWER LIMIT NONZERO
	1	LOWER LIMIT ZERO
8	0	UPPER LIMIT NONZERO
	1	UPPER LIMIT ZERO
7	0	DUAL/RED. COST NONZERO
	1	DUAL/RED. COST ZERO
6	0	LOWER LIMIT EXISTS
	1	LOWER LIMIT DOESN'T EXIST
5	0	UPPER LIMIT EXISTS
	1	UPPER LIMIT DOESN'T EXIST

The next group of 4 bits represents the characteristics for PENALTIES.

BIT PATTERN	UPPER PENALTY	LOWER PENALTY
0 0 0 0	ZERO	ZERO
0 0 0 1	ZERO	NUMBER
0 0 1 0	ZERO	INFINITY
0 0 1 1	NUMBER	ZERO

0 1 0 0	NUMBER	NUMBER
0 1 0 1	NUMBER	INFINITY
0 1 1 0	INFINITY	ZERO
0 1 1 1	INFINITY	NUMBER
1 0 0 0	INFINITY	INFINITY

The rest of the bit permutations are not used.

The last (0 bit) is used by the interpreter to mark the active and nonactive records when the user uses the ACTIVE or DEACTIVE commands to avoid searching of the entire file.

All the above groups of bits are stored together as a 16 bit binary number, which is stored in 2-byte halfword. ATHENA has provisions for the use of 16 bit halfwords representing absolute magnitudes of 0 - 65535, and can extract any component bits of the halfwords as necessary. (In this sense, the usual signed magnitude of IBM/360 halfword integers is ignored.)

d. The (nonzero, noninfinite) number values which must be stored are located immediately after all the information above. If the file represents a problem with M rows and N columns, then location INDEX - where INDEX=(N+M)*3+4+1 - of the SOLFIL array is the first eligible location for storing number values. The value of INDEX is kept in a 2-byte pointer associated with each row and indicates for that row the location of the first value stored. The sequence for storing these numbers for each row is:

ACTIVITY LEVEL, SLACK/INPUT COST, LOWER LIMIT, UPPER LIMIT, DUAL/REDUCED COST, UPPER PENALTY, LOWER PENALTY.

3. SUPERSPARSE Data Structure

This type of data structure takes advantage of the fact that in most problems many number values in the solution file are the same. Each distinct value is stored only once and a 2-byte pointer is used to access this value when needed. This is the only difference from the SPARSE representation (see Figure 3).

The array with the packed solution file is now separated into 3 parts:

a. The first part is exactly the same as in SPARSE.

b. The second part is substantially the same with the following differences:

(1) It consists of 2-byte halfwords instead of 4-byte words.

(2) Each halfword is a pointer to the third part of the array where the distinct number values are stored.

c. The third part consists of a pool of 4-byte words, each representing a distinct real number value. The pointers to the distinct real number values are relative addresses in the real number pool, so a file which is packed with a different array size can be used with the current pointers providing the array size is large enough to hold the file.

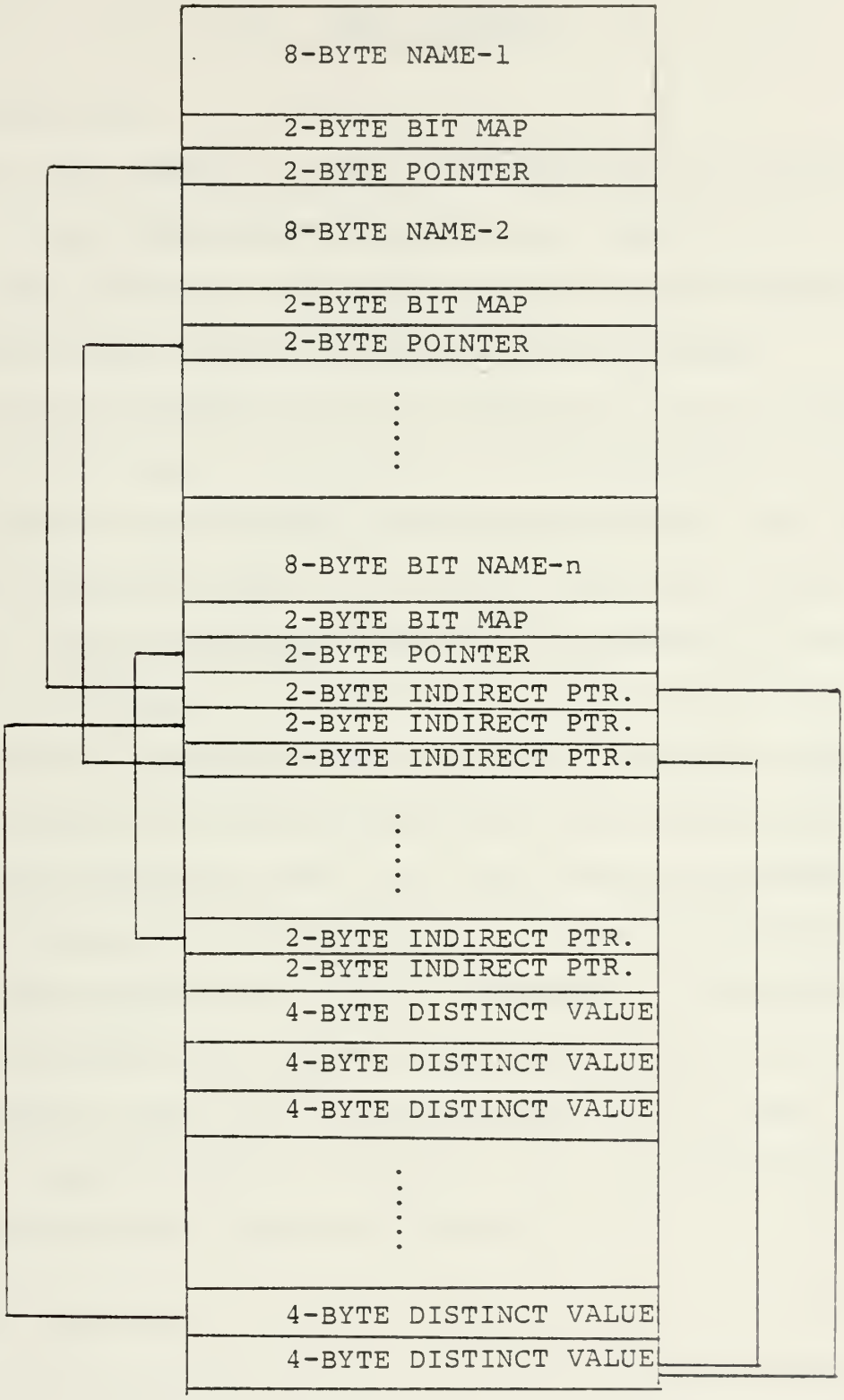


FIGURE 3. SUPERSPARSE Data Structure

III. USER'S MANUAL

A. INTRODUCTION

The system ATHENA is a set of programs which accepts as input a linear programming solution file, packs it in a special data structure and interactively extracts specific information from that file through a set of Queries.

The size of memeory which is required to run the system depends on the size of the file to be accommodated, and thus on the size of the original optimization problem. The user extracts information from the solution file with a Query Language, asking questions related to the solution of the problem represented in the file.

The entire system has been developed in FORTRAN language for portability and better coordination with other Linear Programming procedures which are also written in FORTRAN.

The Queries are self-documenting and their syntax follows closely the syntax of the English language. To avoid typing effort for experienced ATHENA users, short forms of Queries are provided. Only the characters comprising the short forms are interpreted by the system, with all subsequent contiguous nonblank characters ignored.

B. QUERY LANGUAGE

The Query Language consists of three subsets of Queries:

1. The Control Queries:

With this subset of Queries the user controls mainly the output of the system, inserting comments, headings, etc.

2. The Command Queries:

With these the user communicates with the solution file and extracts the specific information he needs.

3. The SET Command.

This command qualifies the ATHENA queries to access only a subset of the problem file.

1. Control Queries

a. VERIFY (Short Form V)

All the following Queries will be displayed with the output. This Control Query is useful when the OFFLINE printer is used for the output instead of the terminal, or when the system is used under Batch Processing; in these cases answers are transmitted to the output device without the corresponding questions if the system is not in VERIFY mode.

b. NOVERIFY (NOV)

The following Queries do not appear with the output. This Control Query is most frequently used when a terminal is used for all output. The DEFAULT mode of the ATHENA system is NOVERIFY.

c. PROMPT (P)

The system responds with the prompt:

' INPUT A COMMAND '

whenever it is ready to accept a Query. PROMPT is a DEFAULT mode of the system.

d. NOPROMPT (NOP)

Used to avoid the prompting phrase in the output, especially when the OFFLINE printer or the Batch Processing is used.

e. H Any Character String

When the first column of a Query is the letter H, then the character string is printed as is in the output. H is used to insert comments or headings in the output.

f. * Any Character String

When the first column of a Query is the character *, no action takes place. This is considered as a comment and is ignored. * is useful to insert comments and/or headings on the terminal output.

g. END (E)

Used to end the current session.

2. Command Queries

A Command Query consists of several fields. Some fields are required and must always appear in a Command Query and others are optional. Each field is separated from the others by at least one blank character. The number of blanks between fields is not significant and a Query may start at any character position in the command. The length of a Query cannot exceed 80 characters including the spaces between the fields.

The possible fields that can be included in a Command Query are:

a. TYPE

b. SELECT

c. MASK

d. CONDITION

e. PRINT OPTION

The fields in a Command Query must appear in the above sequence and the first 3 of them must always appear, with only 2 exceptions. A detailed description follows of each individual field and the way that it may be used.

a. TYPE field

This is the first field of the Query and may start at any character position. This field can be one of the following:

(1) DISPLAY (D)

Used when all the records which meet the requirements of the other fields are to be displayed in the output in the sequence they are encountered starting from the beginning of the solution file.

What portion of each individual record will be displayed depends on the PRINT OPTION field.

(2) COUNT (C)

Used when only the number of records which meet the requirements of the other fields is desired. COUNT is especially useful immediately preceding a DISPLAY command so the user will know in advance the size of output, avoiding

unpredictably extensive printouts. For this Query the PRINT OPTION is ignored as meaningless.

(3) ADD (A)

Used when some numerical quantities of the qualified records are to be summed. The names of the numeric quantities of each record that will be added are given in the PRINT OPTION field. If no PRINT OPTION appears, all the numberic quantities of each record are added and their sums are displayed with appropriate labels.

Since it is mathematically meaningless to add LOWER or UPPER LIMITS, or PENALTIES, they can not be summed or displayed.

(4) AVERAGE (AV)

Used exactly as the ADD command to display arithmetic averages. The sums are divided by the total number of the qualified records.

(5) ACTIVATE (AC) (Syn. ACTIVE)

With the ACTIVATE command the user can indicate a subset of the records of the solution file with specific qualifications determined by the other fields so that subsequent Queries will implicitly refer only to that subset. The user can expand the initial subset by using the ACTIVE command repeatedly to add new records to the active subset.

The command ACTIVATE can minimize the searching time for the required information in the active subset. Each time the ACTIVATE command is issued, the system

responds with the number of records added to the active subset and the current total number of active records.

(6) DEACTIVATE (DE) (Syn. DEACTIVE)

Used to delete records with specific qualifications from the current active subset - created by the ACTIVE commands - or to eliminate any active file. The system responds with the number of records deactivated and the total number of records remaining active.

The entire active file can be deactivated by:

' DEACTIVE ALL or DE A'.

With this Query all the currently active records will be deactivated and the message:

' F I L E D E A C T I V E '

will be printed out. Subsequent queries will refer to the entire solution file.

b. SELECT field

This field is mandatory and specifies whether the qualified records are ROWS, COLUMNS or BOTH. It may consist of one of the following:

(1) ALL (A)

Specifies that the entire file must be searched for the qualified records starting from the first ROW and continuing to the last COLUMN.

(2) COLUMNS (C)

Specifies that the COLUMNS only will be searched for the qualified records starting with the first COLUMN and continuing to the last COLUMN.

(3) ROWS (R)

Specifies that the ROWS only will be searched for the qualified records starting with the first ROW and continuing to the last ROW.

c. MASK field

Specifies that any record is qualified for processing if the name of the record fits the MASK field. The MASK field is left justified and may contain 1 to 8 characters. All the right unfilled positions up to 8 characters are assumed to be the character *. The MASK is matched against the name, starting from the left, character by character. Any character in the name is matched with a * in the MASK field. The MASK field is mandatory.

EXAMPLES

i. The MASK 'X******Y' specifies all the names starting with the letter X and having as the 8th (last) character the letter Y.

ii. The MASK 'X' is equivalent with the MASK 'X*******' and means all the names starting with the letter X.

iii. The MASK '******Y' specifies all the names ending with the letter Y and it is NOT equivalent with the MASK 'Y'.

iv. The MASK 'ABCDEFXY' specifies only this name and since the names of ROWS and COLUMNS are assumed to be inclusively unique, the searching of the file stops when the first match is made.

v. The MASK '*' specifies ALL the names and may be used when no particular mask is desired.

d. CONDITION field

The syntax of this field is:

FOR (<conditional phrase>)

The word FOR, left parenthesis and right parenthesis must always appear when the CONDITION field appears in a Query.

There are two kinds of conditional phrases: The simple conditional phrase and the compound conditional phrase.

(1) Simple Conditional Phrase

There are 3 kinds of simple conditional phrases: The Relational, the Status and the Bound simple conditional phrases.

(a) Relational Simple Conditional Phrase

The syntax is:

<Arg1> <Relop> <Arg2>

where Arg1, Arg2 and Relop are one of the following:

i. Arg1

X	for	ACTIVITY LEVEL
S or C	for	SLACK ACTIVITY or INPUT COST

L	for	LOWER LIMIT
U	for	UPPER LIMIT
D	for	DUAL ACTIVITY or REDUCED COST
P	for	UPPER PENALTY
W	for	LOWER PENALTY

ii. Relop

Relational operators EQ, NE, GT, GE, LT, LE with the same meaning as in FORTRAN. (Note, however, that there are not imbedding decimal characters as in FORTRAN.)

iii. Arg2

Arg2 is defined exactly as Arg1 with the enhancement that Arg2 may also be any integer or real number. Arg2 cannot be expressed as a floating point number in exponential notation.

(b) Status Simple Conditional Phrase

The syntax is:

STATUS <Flag> or ST <Flag>

where Flag is one of the following:

BS	for	BASIC
LL	for	LOWER LIMT
UL	for	UPPER LIMIT
EQ	for	FIXED
VC	for	VOID COLUMN
SC	for	SINGLETON COLUMN

FC	for	FIXED COLUMN
BC	for	BOUND CHANGED
VR	for	VOID ROW
SR	for	SINGLETON ROW
RR	for	REDUNDANT ROW
FR	for	FREE ROW
ER	for	DOUBLETON EQUATION
TR	for	TIGHTEN RANGE

(c) Bound Simple Conditional Phrase

The syntax is:

<Arg1> MINIMUM or <Arg1> MAXIMUM

where Arg1 is specified as in Relational Simple Conditional Phrase. The words MAXIMUM or MINIMUM can be abbreviated as MAX or MIN, respectively. This is used to extract those records which have the MAXIMUM or MINIMUM value in the specified field with the specified MASK. The system responds with the first record encountered with the maximum or minimum value associated with Arg1, and the total number of records that meet the requirements. This phrase may not be used with ACTIVE or DEACTIVE options in the TYPE field of the Query.

(2) Compound Conditional Phrase

The syntax of this phrase is:

<Relational Cond. Phrase> <Log. Oper.> <Relational Cond. Phrase>

or

<Relational Cond. Phrase Log. Oper. Status Cond. Phrase>

where Log. Oper. is OR or AND with the meaning of the corresponding logical operators. Note that the Bound Simple Conditional Phrase is not compatible for use in a Compound Conditional Phrase, since it exhibits no boolean value. Also, the Status Conditional Phrase must always appear after the logical operator.

The CONDITION field as a field must be separated by at least one blank from the other fields of the Query. The word FOR, the left parenthesis, and the first element of the conditional phrase do not require separation by blank characters, nor do the last element of the conditional phrase and the right parenthesis.

The CONDITION field is optional and need not appear in the Query. If it is not present, any record is qualified if the MASK field is satisfied. Using the ACTIVATE and DEACTIVATE commands the user can actually have unlimited length conditional phrases, by adding qualified subsets of records in the ACTIVE file.

e. PRINT OPTION field

This is the last field of a Query. It is optional, and if it does not appear the entire record which satisfy both the MASK and the CONDITION fields are printed out.

The elements of the PRINT OPTION field may be any combination of the following:

X, C or S, L, U, D, P, W

with meanings as described in the CONDITION field. The output will include information described in the PRINT OPTION with corresponding headings. The elements of the field can be separated by any number of blanks, by commas, or not at all. If both C and S appear in the PRINT OPTION neither of them is printed out. For the commands ADD and AVERAGE the default PRINT OPTION is X, C or S, D since there is no meaning for LIMITS and PENALTIES.

For all Queries that potentially require more than one output record for the answer (i.e., all Queries except COUNT, ACTIVE, DEACTIVE and SET), the output will include the following entries for each record:
NUMBER, NAME, STATUS and the entries specified in the PRINT OPTION field in the sequence in which they appear. At the end of the answer output for each Query the total number of qualified records is given. The heading for the output is determined by the SELECT field. If for this field the option ALL is used, the heading will be the one for ROWS although COLUMNS may also be included in the output.

3. The SET Command

By default each time a Command Query is issued the whole solution file is searched starting at the first ROW or

COLUMN and continuing by examining sequentially all the records.

Queries may sometimes apply only to a small part of the solution file or to records whose relative position in the file is known. In these cases the SET command can cause searching to be initiated at a particular entry in the file and continued to another particular entry. Also a fixed step size can be specified for the search. Thus much computational effort can be avoided.

The syntax for the SET Command is:

SET <number 1> <number 2> <number 3>

where;

number 1 is the number of the starting record

number 2 is the number of the record to stop searching

number 3 is the step for searching.

All these numbers must be integers separated by at least one blank and the presence of all of them is required. These numbers also must be in the range of total number of records for the file. The SET limits apply to qualify any subsequent search of the file even if ROW or COLUMN subsets are specified by a Query.

EXAMPLE

Suppose the solution file has 300 rows and 2000 columns and the following SET command is issued:

SET 18 1500 10

For all subsequent Queries:

If the SELECT field of the Query is ALL then the searching starts at the 18th record and continues through the 1500th record with step 10 (i.e., Record numbers 18, 28, 38, ... are examined).

If the SELECT field is ROWS then the searching will start at the 18th row through the last row (300th) with step 10.

If the SELECT field is COLUMNS then the searching will start at 18th column through the 1500th column (or equivalently the 318th record through the 1800th record) since the number of columns is greater than 1500.

To restore default settings, use:

'SET DEFAULT' or 'SET D'

C. ERROR MESSAGES

The following error messages are typed at the terminal as soon as they are detected. If the error is only in syntax, the system is immediately ready to accept a new query, otherwise execution is terminated. Errors have been grouped with one message for each group. Messages are self explanatory.

ERROR NO	POSSIBLE REASON
1	: Attempt to parse a blank query.
101	: Invalid TYPE field. One of the characters D,V,C or blank was expected after A.

102 : Invalid TYPE field. One of the characters O,E was expected after S.

103 : Invalid TYPE field. One of the characters P,V was expected after NO.

104 : Invalid TYPE field. No command starts with the given letter.

201 : Missing character or somewhere in the query there is no space delimiter.

202 : There is no space delimiter.

301 : Invalid SELECT field. SELECT field is missing or there is no space delimiter between TYPE and SELECT fields.

502 : Invalid CONDITION field. the word FOR is missing (the string OR was expected after F), or invalid PRINT field.

503 : Missing left parenthesis in CONDITION field.

504 : Incomplete condition field or missing space delimiter.

505 : Missing right parenthesis in condition field.

506 : Invalid OR logical operator. Character R R was expected after O.

507 : Invalid AND logical operator. The string ND was expected after A.

508 : Invalid logical operator. Only OR and AND are accepted.

511 : Invalid operand for status. The character C or S was expected after B.

512 : Invalid operand for status. The character L was expected after L.

513 : Invalid operand for status. The character L was expected after U.

514 : Invalid operand for status. The character Q or R was expected after E.

515 : Invalid operand for status. The character V, S, F or B was expected before C.

516 : Invalid operand for status. The character V, S, R, F, E or T was expected before R.

517 : Invalid operand for status. The character P was expected before P.

518 : Invalid operand for status. The character I, A, C or D was expected before E.

519 : Missing space delimiter after status operand.

520 : Non recognizable operand for status.

601 : Invalid first operand for relational operator in condition field.

602 : Missing space delimiter in a simple conditional phrase.

603 : Invalid relational operator. The character T or E was expected after G.

604 : Invalid relational operator. The character T or E was expected after L.

605 : Invalid relational operator. The character Q was expected after E.

606 : Invalid relational operator. The character E was expected after N.

607 : Invalid operand in bound conditional phrase. The character N was expected after string MI.

608 : Invalid operand in bound conditional phrase. The string AX was expected after M.

609 : Unrecognizable relational operator in simple conditional phrase.

701 : Invalid print field, or missing word FOR in condition field.

1001 : Error in input data. Unrecognizable status code.

1002 : Error in input data. Data encountered has less than the expected number of rows and columns.

D. LIMITATIONS - EXTENSIONS

As mentioned in the introduction, ATHENA is a direct expansion of the PERUSE system. It includes all the features of PERUSE, except the weighted average command and has the following differences and extensions:

1. ATHENA supports two distinct data structures, each different from that of PERUSE. This was necessary in order to support efficient access to individual records or group of records. The SUPERSPARSE data structure is unique to ATHENA.

2. ATHENA accepts as input a simple file which can be easily obtained from the solution file of any linear programming package on tape, disk or cards.

3. ATHENA supports the commands SET, ACTIVATE, DEACTIVATE and COUNT, in addition to the commands of PERUSE, allowing the user to construct logical subsets of the solution and efficiently access these subsets as independent files with very small access time.

4. ATHENA supports compound conditional phrases for extraction of more specific information and the bound conditional phrase for maximum and minimum values.

5. ATHENA uses object time variable format allowing better appearance of output and uses the words NONE and INFINITY instead of the number 0.7273E76 for better readability.

6. ATHENA accepts reduced problems from PREP [2] and can be used to pass the PREP status file with the solution file of any optimization system to permit recovery of the original problem solution.

ATHENA has been designed to handle solution files with up to 30,000 records. The actual limit is imposed by the number of real number values that must be stored explicitly. This number cannot presently exceed 65536 since this is the largest integer pointer value which can be stored by ATHENA in a 2-byte halfword. Experience has shown that the average number of stored values for each record, excluding penalties

is 1.5 [7]. Adding to this another 0.5 per record for penalties, the problem size limit may be as large as 30,000 rows and columns.

A rough estimation of the space needed for the packed file in 4-byte words can be obtained by multiplying the sum of rows and columns of the solution file by 5 for the SPARSE data structure and by 4 for SUPERSPARSE. Before using ATHENA, adjust the size of the SOLFIL array in common block SOLPAC to this number. To avoid passing problems with common areas under the IBM 360/67 Operating System, use an array size that is an exact multiple of 4096 larger than the number calculated above. Also make corresponding adjustments to the DEFINE FILE statement of the main program. Other computers will require analogous modifications to this array size.

ATHENA has been developed in modular form and can be easily changed or extended to support future needs. Commands which can be easily implemented include the weighted average, the sort of output, or further calculations needed for the analysis of the solution file. ATHENA can also be used as part of an integrated system for sensitivity analysis of optimization problems.

E. SYSTEM INTERFACE

1. Preview

Linear Programming packages give differing forms of output so that it is difficult for a system to be interfaced

with all of these solution formats. ATHENA accepts as input a solution file in a 'standard' form which can be easily obtained from any other solution file form.

ATHENA is best utilized in an interactive system such as CP/CMS, although it can also be used in batch processing. On the other hand, most L.P. packages run only in batch processing. Moreover, in some systems there is no integration of interactive and batch processing (as is currently the case at the Naval Postgraduate School Computer Center). In these cases, the solution files may be transferred manually from one system to the other using magnetic tapes or cards.

A simple input file has been designed which can be punched in cards or entered on tape, disk, or other storage media.

2. Input File

The input file consists of records with the following structure:

a. The first record always contains the number of ROWS, the number of COLUMNS, and in position 51 the character '1' if each ROW record contains PENALTIES or '0' otherwise. The FORMAT of the first record is

(I5,30X,I5,10X,I1).

b. Each subsequent record contains explicitly all the information associated with each ROW and COLUMN, with

the following format:

NAME Format 2A4 (left justified)

STATUS Format A2

X, C or S, L, U, and D numeric field values with Format 5E14.5 or 5F14.5. (The meanings of each of these fields is described in the previous section.) If the solution file includes PENALTIES, then two records will be associated with each ROW. The first will be exactly that described above, and the second will have the FORMAT(E14.5,16X,E14.5) for P, and W, the UPPER and LOWER PENALTIES. In all cases, INFINITE values will be represented explicitly by the number ±0.1E76. The total number of records must agree with the sum of ROWS plus COLUMNS, with the records of ROWS preceeding those of COLUMNS; otherwise an Input error will occur. The file is read in and packed one record at a time.

3. Packed File

a. Packed File as Input

If the input file is already packed from a previous use of the system, it will be read in unformated binary format. The system will provide the user information for memory requirements before reading the file. The packed file may be on a tape or disk but cannot be on cards. The system will ask the user at the beginning of a session for the number of the file.

b. Packed File as Output

If an unpacked file is used as input, the system will ask for the file number where a packed file is to be

written. Of course, a DEFINE FILE FORTRAN statement must be included right after the declarations of the main program.

E. EXAMPLE OF SYSTEM USE

The procedure follows for use of ATHENA at the Naval Postgraduate School Computer Center with the IBM 360/67. The solution file here is produced by the MPS/360 package and ATHENA is used under CP/CMS. Similar procedures can be followed for any other installation.

1. Obtaining the Unpacked Solution

a. Submit the problem to be solved using the usual Control Cards required for the MPS/360 package inserting before the Control Card:

```
//MPS2.SYSIN DD *
```

the following cards:

```
//MPS2.SYSPRINT DD DSN=Sxxxx.nnnnnn,
// UNIT=3330,VOL=SER=DISK04,
// SPACE=(CYL,(1,1)),DISP=(NEW,KEEP),
//  DCB=(RECFM=UA,BLKSIZE=133)
```

With these cards the output of MPS will go to the DISK instead of the printer.

xxxx is the user's number and
nnnnnn is the file name on the disk.

If the solution file is too big or disk space is not available use tape or tapes to store the output.

b. Use the program REWRITE (page 98) to transform the MPS/360 tape or cards to the format required for the ATHENA unpacked Input file.

c. Now the unpacked file for ATHENA is available and can be used to analyze the solution.

2. Using the System Under CP/CMS

ATHENA in CP/CMS TEXT form requires about 67K bytes. The space needed for the packed file depends on the number of records, the method used for packing and the density of the original file. 200K bytes would be sufficient to hold a packed file with up to 13,000 rows and columns. After sufficient space has been secured, the following procedure may be applied:

a. Ask OPERATOR to connect the tape with the Input file created by the REWRITE to the private disk as device 181. As soon as the tape is connected, the message 'DEVICE 181 ATTACHED' will be printed at the terminal.

b. Before using the tape, type ALWAYS under CMS the command ' TAPE SKIP 1 '. This command will position the tape at the first record of the file. This command is required because the tape created by IBM O.S./360.

c. Type $ ATHENA

ATHENA will ask for information about the file identifiers for input-output, whether the file is packed and method of packing and will give the size of the packed

file. For input file ordinal use any number between 01 and 99 excluding the numbers 03 - 06. For output file ordinal give the number 03 or 04. These are the numbers used by the DEFINE FILE FORTRAN statement and they can be changed. The system will be ready to accept QUERIES as soon as the prompt phrase 'INPUT A COMMAND' is typed by ATHENA at the terminal. The packed file can be saved on a tape using the 'TAPE DUMP' command under CMS.

In the next few pages a demonstration of using ATHENA with a solution file of 766 rows and 10921 columns is given. This problem is a mixed integer optimization model with 963 binary variables for medium term capital budgeting of the Naval Air Test Center [6]. For this problem, a query may require as much as one and a half minutes of clock time if the interactive system on the IBM 360/67 is under heavy yes and the query is difficult to answer. However, most queries are answered almost immediately. Response time is especially good when the user makes use of ACTIVATE, MASK and SET features to qualify necessary searching.

```
$ ATHENA
 WHAT IS THE FILE NO OF THE SOLUTION FILE ? (FORMAT I2)
04
 IS THE FILE ALREADY PACKED ? ENTER YES OR NO :
YES
    MEMORY REQUIRMENTS  FOR THE  PACKED FILE  :47749  4-BYTE
WORDS

  IF YOU HAVE SUFFICIENT MEMORY SPACE ENTER YES

  OTHERWISE ENTER  NO  MAKE  ADJUSTMENTS AT  COMMON AND  TRY
AGAIN
YES
                  INPUT A COMMAND
**************************************************************
*   USING THE CHARACTER * AS THE FIRST COLUMN OF THE QUERY   *
*   THE QUERY IS IGNORED AND THIS IS A CONVENIENT WAY TO     *
*   INSERT COMMENTS IN THE OUTPUT FROM A TERMINAL.           *
*   COMMENTS IN THE OUTPUT FROM THE OFFLINE PRINTER ARE      *
*   INSERTED USING THE LETTER H INSTEAD OF *.                *
*   TO AVOID THE PROMPTNESS PHRASE 'INPUT A COMMAND' AT      *
*   THE OUTPUT USE THE COMMAND 'NOPROMPT'.                   *
*   BECAUSE OF THE REQUIREMENTS FOR THE APPEARANCE OF THE    *
*   THESIS THE PRINT OPTION FIELD OF THE DISPLAY COMMAND     *
*   IS USED ONLY WITH AT MOST TWO OPTIONS.                   *
**************************************************************
*
*
*   HERE IS A DEMONSTRATION OF USING THE SYSTEM ATHENA.
*
*
*   HOW MANY ROWS HAS THE FILE ?
*
COUNT ROWS *

   766  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   HOW MANY COLUMNS ?
*
COUNT  COLUMNS  *

 10921  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   HOW MANY OF THEM ARE BASIC ?
*
   C   C   *  FOR(ST BS)
```

```
   506  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   HOW MANY OF THEM ARE EQUAL ZERO ?
*
   C C * FOR( X EQ 0. AND ST BS)

    55  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   USING THE COMMAND 'ACTIVATE' ONLY THE ACTIVATED
*   RECORDS ARE SEARCHED TO ANSWER THE QUESTION.
*   THIS IS A GOOD WAY TO AVOID SEARCH OF THE WHOLE
*   FILE.
*
ACTIVATE  COLUMNS  XO

        73   RECORDS ACTIVATED
     TOTAL RECORDS ACTIVE :     73
*
*   HOW MANY ROWS NOW ?
*
  C R *

     0  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   THERE ARE NO ROWS SINCE ONLY COLUMNS ACTIVATED.
*   HOW MANY OF THEM ARE BASIC ?
*
  C C * FOR(ST BS)

     1  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   DISPLAY THEM
*
   DISPLAY C * FOR(ST BS) X L
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER  .NAME..  AT  ...ACTIVITY...  ..LOWER LIMIT.
  803  X041      BS        0.12910         0.00000

     1  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   WHAT IS THE AVERAGE OF NONZERO ACTIVITIES ?
*
AVERAGE C * FOR(X NE 0)
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS
```

```
NUMBER  .NAME..  AT  ...ACTIVITY...  ..INPUT COST..
 SUMS OR AVERAGES :        0.98773          0.76576


    71  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   THE SUM OF ACTIVITIES AND INPUT COST ?
*
ADD C * FOR ( X NE 0 )
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER  .NAME..  AT  ...ACTIVITY...  ..INPUT COST..
 SUMS OR AVERAGES :       70.12909         54.36882


    71  ROWS OR COLUMNS WITH MASK : ********


 SATISFY THE CONDITIONS
*
*   ADD SOME NEW COLUMNS AT THE ACTIVE FILE
*
ACT  C  YM01

       175   RECORDS ACTIVATED
     TOTAL RECORDS ACTIVE :   248
*
*   BASIC ?
*
C C * FOR (ST BS)

     7  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   DISPLAY BASIC AND ZERO ACTIVITY
*
  D  C  * FOR ( X EQ 0 AND STATUS BS)  X C
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER  .NAME..  AT  ...ACTIVITY...  ..INPUT COST..
 5568  YM01061   BS         0.00000         -0.01000


     1  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   HOW MANY INPUT COSTS ARE ZERO ?
*
  C C * FOR( C EQ 0 )
```

```
     5  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   HOW MANY AT LOWER LIMIT ?
*
  C C * FOR(STATUS LL)

   171  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   ELIMINATE SOME RECORDS FROM THE ACTIVE FILE
*
  DEACTIVATE C X


        73   RECORDS  DEACTIVATED
     TOTAL RECORDS  ACTIVE :   175
*
*   CHECK FOR THE ELIMINATION
*
  C C X

     0  ROWS OR COLUMNS WITH MASK : X********
 SATISFY THE CONDITIONS
*
*   ELIMINATION O.K.
*

*   HOW MANY AT LOWER LIMIT (LL) NOW ?
*
C C * FOR (ST LL)

   169  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   WHAT IS THERE AVERAGE ?
*
  AV C * FOR(ST LL)
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER  .NAME..  AT  ...ACTIVITY...   ..INPUT COST..
 SUMS OR AVERAGES :         0.00000         -0.01000


   169  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   RETURN TO THE ORIGINAL BIG FILE
*
DEACTIVATE  ALL
          F I L E   D E A C T I V E
```

```
*
*   DISPLAY THE 5 FIRST ROWS
*
  SET 1 5 1
  D R *  X S
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER   .NAME..   AT   ...ACTIVITY...   SLACK ACTIVITY
     1   BENEFITS   BS        449.49902       -449.49902
     2   RC1011     UL          0.00000          0.00000
     3   RC1021     UL          0.00000          0.00000
     4   RC1031     UL          0.00000          0.00000
     5   RC1041     BS          0.00000          0.00000

      5  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   THE FIRST 5 COLUMNS
*
  D C *  C D
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER   .NAME..   AT   ..INPUT COST..   .REDUCED COST.
   767   X001       UL          0.99994          0.36312
   768   X002       UL          0.89108          0.77450
   769   X003       UL          0.45629          0.23338
   770   X004       UL          0.45483          0.36919
   771   X005       UL          0.73440          0.60813

      5  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   RETURN TO THE ORIGINAL FILE
*
SET DEFAULT
*
*   DISPLAY VARIABLES IN A SPECIFIC RANGE
*
 D C YM  FOR (X GT 10.5 AND X LT 25.5) X L
 THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS

NUMBER   .NAME..   AT   ...ACTIVITY...   ..LOWER LIMIT.
 5708   YM05061    BS         14.00000          0.00000
 6139   YM17171    BS         11.54310          0.00000
 6247   YM20201    BS         17.13029          0.00000
 6758   YM35061    BS         17.29999          0.00000
 7472   YM20202    BS         18.98279          0.00000
 7983   YM35062    BS         11.90000          0.00000
 8697   YM20203    BS         21.00000          0.00000
 9208   YM35063    BS         16.59999          0.00000
 9378   YM05014    BS         11.90000          0.00000
 9922   YM20204    BS         13.00000          0.00000
10433   YM35064    BS         23.06062          0.00000
10608   YM05065    BS         14.00000          0.00000
11147   YM20205    BS         13.00000          0.00000
11658   YM35065    BS         19.50000          0.00000
```

```
    14  ROWS OR COLUMNS WITH MASK : YM******
 SATISFY THE CONDITIONS
*
*   WHAT IS THE MAXIMUM OF THE ACTIVITY LEVEL ?
*
  D  C  *  FOR(X MAX) X C
   1  YM18184   BS         88.63869          0.00000


     1  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   THE MINIMUM LOWER LIMIT FOR THE ROWS ?
*
  D  R  *  FOR(L MIN) L U
 766  BENEFITS  BS              NONE              NONE


   766  ROWS OR COLUMNS WITH MASK : ********
 SATISFY THE CONDITIONS
*
*   THE MAXIMUM ACTIVITY LEVEL FROM VARIABLES
*   STARTING WITH LETTER Y ?
*
D C Y FOR ( X  MAX )   XC
   1  YM18184   BS         88.63869          0.00000


     1  ROWS OR COLUMNS WITH MASK : Y*******
 SATISFY THE CONDITIONS
*
*   ARE THERE INFEASIBILITIES ?
*
COUNT   ALL  *   FOR (ST IN)

     0  ROWS OR COLUMNS WITH MASK : ********

 SATISFY THE CONDITIONS
*
*   EXIT FROM THE SYSTEM
*
       END
```

APPENDIX

The whole program consists of 3 main parts: The input part, the parser-code generation part and the interpreter part (see Figure 1). The main program calls these 3 parts according to the instructions from the user. More detailed information for the main subroutines of each part are given below.

1. Input Part.

There are 2 almost identical subroutines named READF and READD. READF is used to pack the solution file in SPARSE data structure and READD in SUPERSPARSE.

a. Input Parameters

(1) IFLR: The file number of the solution file. This can be any number between 1 - 99 excluding the numbers 3, 4, 5, and 6.

(2) IFLW: The file number where the packed file will be written. Currently this number can be 3 or 4 since the DEFINE FILE statement in the main program defines the files 3 and 4 only.

b. Output Parameters

(1) NROWS, NCOL; The number of rows and columns of the solution file respectively.

(2) IER: If this value is different from zero an error has been encountered in the input data.

The solution file is read in standard format, as explained in user's manual, one record at a time, filling the array SOLFIL in the common area SOLPAC with the packed file. The name of the record is read directly into SOLFIL and the associated values into the array DATA. The 2 character status code is read into the INTEGER*2 variable STAT. The INTEGER*2 array CSTAT has all possible values for status, and their location in this array, minus one, gives the corresponding value of the first bit group in the 16 bit map as explained in the data structure section.

The variable INDEX always points to the next available location for storing the next nonzero, noninfinite number. The value of INDEX at the beginning of the processing of each record is stored in the 2-byte pointer after the name. Each number in the array DATA is examined and if it is eligible for storage it is stored at the location SOLFIL(INDEX) and INDEX is increased by one. At this point, for the subroutine READD, the end of the array SOLFIL, where the distinct real number values are stored, is searched and if the value sought is found the location of this value, counting from the end of the array is stored at the location SOLFIL(INDEX); otherwise, a new entry is made and the location of this entry is stored at the location SOLFIL(INDEX). Notice that SOLFIL(INDEX) is a 4-byte location for READF but 2-byte for READD as explained earlier. After the last record has been packed, READD shifts the distinct real number values

down to the end of 2-byte indirect pointers. When the system is to be used under batch processing the CALL TAPSET statement from READF and READD must be removed and the input file must be described using the usual DD statement.

2. Parser - Code Generator

There are several subroutines in this part of the system. There is one subroutine for each field of the query, the control subroutine PARSE and various others to perform specific tasks like getting the numerical value of a string of decimal characters,etc. The subroutine PARSE is described below. The numerical codes used for each case have been inserted as comments at the beginning of each subroutine where they are used.

a. Input Parameters

(1) QRY: INTEGER*2 array with the query to be parsed in A format.

b. Output Parameters

(1) CODE: INTEGER*2 array with the numerical code for query as follows:

QRY(1)	: TYPE	field
QRY(2)	: SELECT	field
QRY(3)	: MASK	field
QRY(4) - QRY(10)	: CONDITION	field
QRY(11) - QRY(18)	: PRINT	field

(2) MASK: INTEGER*2 array with the mask from the query, one character per location.

(3) ARG2, ARG4 : numerical values from the condition field, if any.

(4) IER : Code number for syntax error. If IER=0 the query has been parsed correctly.

Subroutine PARSE calls sequentially the corresponding subroutines to parse each field of the query. Before calling it positions the pointer ICOL1 at the location of the QRY where the field starts using the subroutine GETNCH.

3. Interpreter Part

There are several subroutines called by the main subroutine INTERP of this part.

a. Input Parmaeters

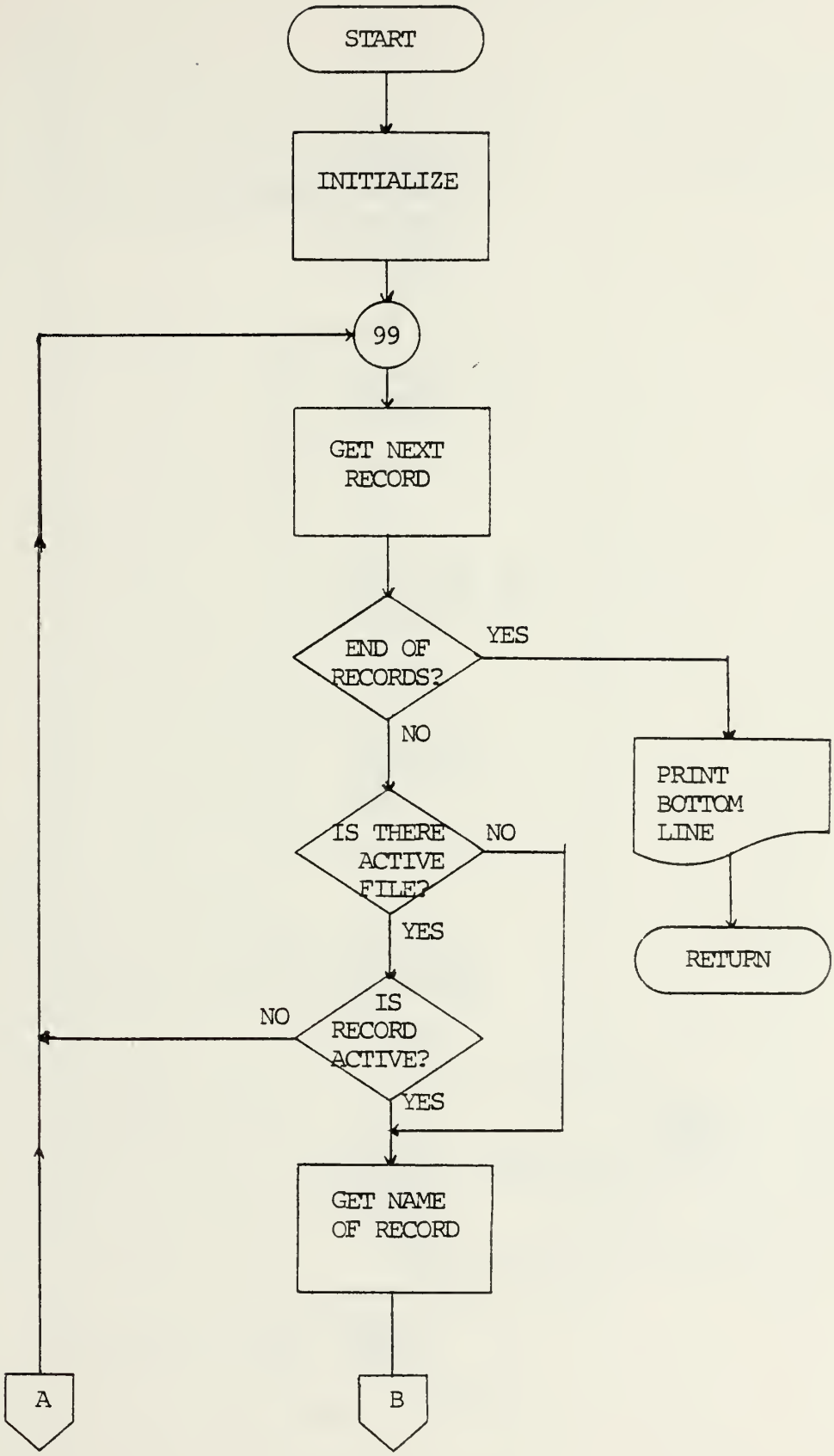
(1) CODE, MASK, ARG2, ARG4, NROWS, NCOL, IER as described previously.

(2) K1, K2, K3 : Record number to start searching, to stop searching and searching step respectively. Default values are 1, (NROWS + NCOL), 1.

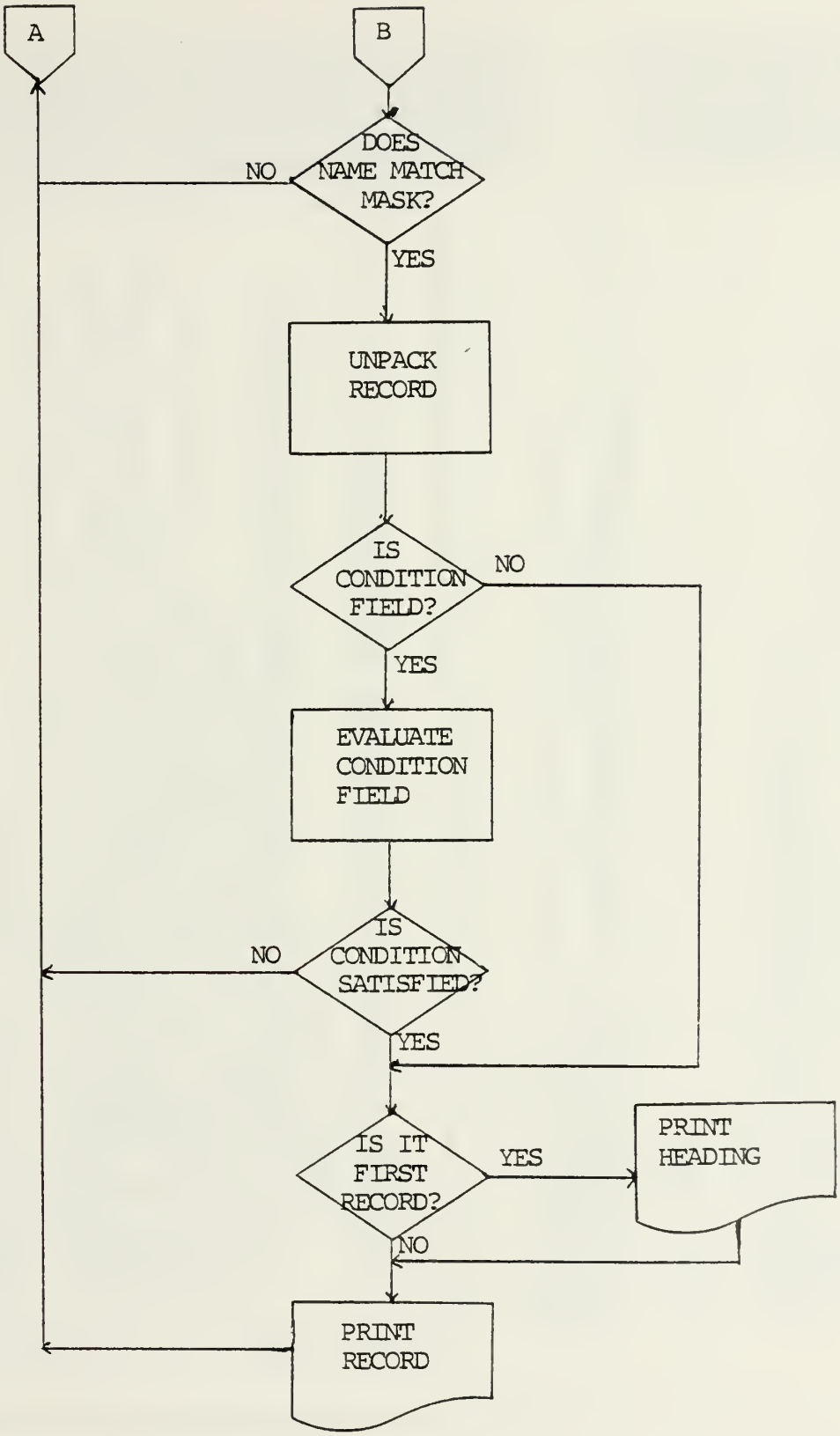
(3) IFLG : Flag to determine the data structure type of the packed file. If IFLG=0, data structure SPARSE otherwise SUPERSPARSE.

(4) IACT : Passed through the INIT common area and, if 0, the searching is done as determined by K1, K2, K3, otherwise there is an active file created by the user through the ACTIVATE command and only the members of this file are searched.

The logic of this subroutine is shown in the flowchart on the next two pages.



INTERPRETER


A
B
DOES NAME MATCH MASK?
NO
YES
UNPACK RECORD
IS CONDITION FIELD?
NO
YES
EVALUATE CONDITION FIELD
IS CONDITION SATISFIED?
NO
YES
IS IT FIRST RECORD?
YES
PRINT HEADING
NO
PRINT RECORD

```
C    ************************************************************** MAI00010
C    *                                                            * MAI00020
C    *          GALATAS  PANAGIOTIS / HELLENIC ARMY               * MAI00030
C    *                  - MARCH   1979 -                          * MAI00040
C    *                                                            * MAI00050
C    *  NAME OF THE SYSTEM : ATHENA                               * MAI00060
C    *  LANGUAGE           : FORTRAN G                            * MAI00070
C    *                                                            * MAI00080
C    *     SYSTEM ATHENA IS USED FOR HANDLING A LARGE SCALE LINEAR * MAI00090
C    *  PROGRAMMING SOLUTION FILE.                                * MAI00100
C    *  USES AS INPUT A SOLUTION FILE IN STANDARD FORMAT AND THROUGH * MAI00110
C    *  A SET OF QUERIES HELP THE ANALYST TO GET INTERACTIVELY THE * MAI00120
C    *  SPECIFIC INFORMATION HE NEEDS TO ANALYZE THE SOLUTION WITHOUT * MAI00130
C    *  GOING THROUGH THE STACK OF COMPUTER OUTPUT.               * MAI00140
C    *     THE SYSTEM PACKS THE SOLUTION FILE INTO A SOPHISTICATED * MAI00150
C    *  DATA STRUCTURES FOR MEMORY SPACE ECONOMY.                 * MAI00160
C    *  TWO KIND OF DATA STRUCTURES ARE EMPLOYED :                * MAI00170
C    *    1. FOR GENERAL TYPE SOLUTION FILE                       * MAI00180
C    *    2. FOR SOLUTION FILES WHERE THE NUMBER OF DISTINCT VALUES * MAI00190
C    *       IS RELATIVELY SMALL (LESS THAN ONE HALF OF THE TOTAL * MAI00200
C    *       NUMBERS STORED.)                                     * MAI00210
C    *                                                            * MAI00220
C    *     FOR MORE INFORMATION ABOUT ACCEPTABLE QUERIES, DATA    * MAI00230
C    *  STRUCTURES AND USE OF THE SYSTEM SEE USER'S MANUAL.       * MAI00240
C    *                                                            * MAI00250
C    *                                                            * MAI00260
C    ************************************************************** MAI00270
C                                                                   MAI00280
C                M A I N     P R O G R A M                       MAI00290
C                                                                   MAI00300
C     COPYRIGHT  1979
C     PAN. GALATAS, GREEK ARMY HEADQUARTERS, ATHENS, GREECE
C     ALL RIGHTS RESERVED
C     REPRODUCTION OF ALL, OR ANY PART OF SYSTEM ATHENA
C     IS PROHIBITED WITHOUT WRITTEN PERMISSION OF THE AUTHOR.
C                                                                   MAI00310
C                                                                   MAI00320
C    DECLARATIONS                                                   MAI00330
C                                                                   MAI00340
C                                                                   MAI00350
      COMMON ITM(6),TIME                                           MAI00360
      INTEGER*2 QRY(80),MASK(8),CODE(20)                           MAI00370
      COMMON /INIT/ZERO,TINF,IACT,MEMOR,ISIZE                      MAI00380
      COMMON /SOLPAC/SOLFIL(81920)                                 MAI00390
      COMMON /CHAR/A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,     MAI00400
     -BLANK,COMMA,STAR,LPAR,RPAR                                   MAI00410
      INTEGER*2 A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,       MAI00420
     -BLANK,COMMA,STAR,LPAR,RPAR                                   MAI00430
```

```
      REAL ARG2,ARG4                                                    MAI00440
      LOGICAL*1 VER,PROM,HEAD                                           MAI00450
      DATA KK/'NO  '/                                                   MAI00460
      DEFINE FILE 4(27,1775,U,II)                                       MAI00470
      DEFINE FILE 3(32,1775,U,JJ)                                       MAI00480
C
CC    OBJECT CODE COPYRIGHT STATEMENT
C
      IF(.FALSE.) WRITE(6,9999)
9999  FORMAT( '1COPYRIGHT 1979',
     1 'PAN.  GALATAS, GREEK ARMY HEADQUARTERS, ATHENS, GREECE',/,
     2 'OALL RIGHTS RESERVED')
C
      MEMOR=81920                                                       MAI00490
      CALL IXCLOK(ITM)                                                  MAI00500
      TIME=ITM(5)/76800.                                                MAI00510
C                                                                       MAI00520
CC                                                                      MAI00530
CC  GET INFORMATION NEEDED FOR THE SOLUTION FILE                        MAI00540
CC                                                                      MAI00550
C                                                                       MAI00560
      WRITE(6,1004)                                                     MAI00570
1004  FORMAT(2X,'WHAT IS THE FILE NO OF THE SOLUTION FILE ? (FORMAT I2)'MAI00580
     -)                                                                 MAI00590
      READ(5,1005) IFLR                                                 MAI00600
1005  FORMAT(I2)                                                        MAI00610
      WRITE(6,1001)                                                     MAI00620
1001  FORMAT(2X,'IS THE FILE ALREADY PACKED ? ENTER YES OR NO :')       MAI00630
      READ(5,1002) K1                                                   MAI00640
1002  FORMAT(A4)                                                        MAI00650
      IF(K1.NE.KK) GO TO 1099                                           MAI00660
      WRITE(6,1003)                                                     MAI00670
1003  FORMAT(2X,' FOR BETTER MEMORY UTILIZATION ANSWER YES OR NO TO THE MAI00680
     -FOLLOWING QUESTION :',//,1X,'DO YOU EXPECT THE DISTINCT VALUES OF MAI00690
     -THE SOLUTION FILE TO BE LESS THAN HALF OF THOSE NUMBERS EXCLUDING MAI00700
     -ZEROES AND INFINITY ? :')                                         MAI00710
      READ(5,1002) K1                                                   MAI00720
      WRITE(6,1007)                                                     MAI00730
1007  FORMAT(5X,'WHAT IS THE FILE NO YOU WANT THE PACKED FILE TO BE WRITMAI00740
     -TEN ? (FORMAT I2) ')                                              MAI00750
      READ(5,1005) IFLW"                                                MAI00760
      WRITE(6,1006)                                                     MAI00770
1006  FORMAT(10X,'READING AND PACKING THE SOLUTION FILE')               MAI00780
C                                                                       MAI00790
CC   READING AND PACKING THE SOLUTION FILE                              MAI00800
C                                                                       MAI00810
      IF(K1.EQ.KK) CALL READF(NROWS,NCOL,IER,IFLR,IFLW)                 MAI00820
      IF(K1.NE.KK) CALL READD(NROWS,NCOL,IER,IFLR,IFLW)                 MAI00830
```

```
      IF(IER.NE.0) GO TO 999                                                    MAI00840
      GO TO 1020                                                                MAI00850
1099  CONTINUE                                                                  MAI00860
      READ(IFLR'1) (SOLFIL(IK),IK=1,100)                                        MAI00870
      IK=SOLFIL(1)                                                              MAI00880
      WRITE(6,1066) IK                                                          MAI00890
1066  FORMAT(5X,'MEMORY REQUIRMENTS FOR THE PACKED FILE :',I5,'  4-BYTE MAI00900
     -WORDS',//,3X,'IF YOU HAVE IT ENTER YES',//,3X,'OTHERWISE ENTER NO MAI00910
     - MAKE ANDJUSTMENTS AT COMMON AND TRY AGAIN')                              MAI00920
      READ(5,1002) K1                                                           MAI00930
      IF(K1.EQ.KK) GO TO 99                                                     MAI00940
      READ(IFLR'1) (SOLFIL(IL),IL=1,IK)                                         MAI00950
1020  NROWS=SOLFIL(3)                                                           MAI00960
      NCOL=SOLFIL(4)                                                            MAI00970
      ISIZE=SOLFIL(1)                                                           MAI00980
      IFLG=SOLFIL(2)                                                            MAI00990
C                                                                               MAI01000
CC                                                                              MAI01010
C   INITIALIZE  FLAGS  -  CALCULATE  LIMITS                                     MAI01020
CC                                                                              MAI01030
C                                                                               MAI01040
      IER=0                                                                     MAI01050
      VER=.FALSE.                                                               MAI01060
      PROM=.TRUE.                                                               MAI01070
      HEAD=.FALSE.                                                              MAI01080
92    K1=1                                                                      MAI01090
      K2=NROWS+NCOL                                                             MAI01100
      K3=1                                                                      MAI01110
10    CONTINUE                                                                  MAI01120
      DO 2 J=1,18                                                               MAI01130
2     CODE(J)=0                                                                 MAI01140
      CALL TIMER                                                                MAI01150
      IF(PROM) WRITE(6,100)                                                     MAI01160
100   FORMAT(15X,' INPUT A COMMAND ')                                           MAI01170
C                                                                               MAI01180
CC  READ THE QUERY                                                              MAI01190
C                                                                               MAI01200
      READ(5,200) QRY                                                           MAI01210
CC                                                                              MAI01220
CC  CHECK  IF  IT  IS  HEADING                                                  MAI01230
C                                                                               MAI01240
      IF(QRY(1).EQ.H) GO TO 61                                                  MAI01250
      IF(VER) WRITE(6,300) QRY                                                  MAI01260
200   FORMAT(80A1)                                                              MAI01270
C                                                                               MAI01280
CC  CALL THE PARSER TO PARSE THE QUERY                                          MAI01290
C                                                                               MAI01300
      CALL PARSE(QRY,CODE,MASK,ARG2,ARG4,ICOL1,IER)                             MAI01310
```

```
      IF(IER.NE.0) GO TO 999                                                    MAI01320
C                                                                               MAI01330
C                                                                               MAI01340
C   CHECK  WHAT  KIND OF QUERY I S IT                                           MAI01350
C   IF IT IS CONTROL PROCESS IT HERE  OTHERWISE CALL INTERPRETER                MAI01360
C                                                                               MAI01370
C                                                                               MAI01380
      IF(CODE(1).GT.10) GO TO 20                                                MAI01390
      IF(CODE(1).EQ.10) GO TO 10                                                MAI01400
      IF(CODE(1).NE.1)  GO TO 30                                                MAI01410
      VER=.TRUE.                                                                MAI01420
      GO TO 10                                                                  MAI01430
30    IF(CODE(1).NE.2)  GO TO 40                                                MAI01440
      VER=.FALSE.                                                               MAI01450
      GO TO 10                                                                  MAI01460
40    IF(CODE(1).NE.3)  GO TO 50                                                MAI01470
      PROM=.TRUE.                                                               MAI01480
      GO TO 10                                                                  MAI01490
50    IF(CODE(1).NE.4)  GO TO 60                                                MAI01500
      PROM=.FALSE.                                                              MAI01510
      GO TO 10                                                                  MAI01520
60    IF(CODE(1).NE.5)  GO TO 70                                                MAI01530
61    QRY(1)=BLANK                                                              MAI01540
      WRITE(6,300) QRY                                                          MAI01550
300   FORMAT(1X,80A1)                                                           MAI01560
      GO TO 10                                                                  MAI01570
70    IF(CODE(1).EQ.6)  GO TO 99                                                MAI01580
      IF(CODE(1).NE.7)  GO TO 80                                                MAI01590
      CALL GETNCH(QRY,ICOL1,BLANK,IER)                                          MAI01600
      IF(IER.NE.0) GO TO 999                                                    MAI01610
      IF(QRY(ICOL1).EQ.D) GO TO 92                                              MAI01620
      MET=0                                                                     MAI01630
90    MET=MET+1                                                                 MAI01640
      DO 1 J=ICOL1,80                                                           MAI01650
      ITEMP=J                                                                   MAI01660
      IF(QRY(J).EQ.BLANK) GO TO 91                                              MAI01670
1     CONTINUE                                                                  MAI01680
91    IF(MET.GT.3) GO TO 10                                                     MAI01690
      ITEMP=ITEMP-1                                                             MAI01700
      IF(MET.EQ.1) K1=VAL(QRY,ICOL1,ITEMP)                                      MAI01710
      IF(MET.EQ.2) K2=VAL(QRY,ICOL1,ITEMP)                                      MAI01720
      IF(MET.EQ.3) K3=VAL(QRY,ICOL1,ITEMP)                                      MAI01730
      IF(K2.GT.(NROWS+NCOL)) K2=NROWS+NCOL                                      MAI01740
      ICOL1=ITEMP                                                               MAI01750
      CALL GETNCH(QRY,ICOL1,BLANK,IER)                                          MAI01760
      GO TO 90                                                                  MAI01770
80    IER=1                                                                     MAI01780
      GO TO 999                                                                 MAI01790
```

```
20     CALL INTERP(CODE,MASK,ARG2,ARG4,K1,K2,K3,NROWS,NCOL,BLANK,STAR,IERMAI01800
      -,IFLG)                                                            MAI01810
       IF(IER.NE.0) GO TO 999                                            MAI01820
       GO TO 10                                                          MAI01830
999    CALL ERROR(IER)                                                   MAI01840
       IF(IER.EQ.0) GO TO 10                                             MAI01850
99     STOP                                                              MAI01860
       END                                                               MAI01870
C                                                                        MAI01880
CC                                                                       MAI01890
CCC    ***************************************************************** MAI01900
CCC    *                                                               * MAI01910
CCC    *  BLOCK  DATA  INITIALIZES  LABELED COMMON  AREAS               * MAI01920
CCC    *                                                               * MAI01930
C      ***************************************************************** MAI01940
       BLOCK DATA                                                        MAI01950
       COMMON /INIT/ZERO,TINF,IACT,MEMOR,ISIZE                           MAI01960
       COMMON /SOLPAC/SOLFIL(81920)                                      MAI01970
       COMMON /CHAR/A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,         MAI01980
      -BLANK,COMMA,STAR,LPAR,RPAR                                        MAI01990
       INTEGER*2 A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,            MAI02000
      -BLANK,COMMA,STAR,LPAR,RPAR                                        MAI02010
       DATA ZERO,TINF,IACT,MEMOR/0.1E-3,0.1E76,0,81920/                  MAI02020
       DATA  A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,                MAI02030
      -BLANK,COMMA,STAR,LPAR,RPAR                                        MAI02040
      -/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HL,1HM,1HN,1HO,1HP,         MAI02050
      -1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1H ,1H,,1H*,1H(,1H)/              MAI02060
       END                                                               MAI02070
C                                                                        PAR00010
CC                                                                       PAR00020
CCC    ***************************************************************** PAR00030
CCC    *                                                               * PAR00040
CCC    *      SUBROUTINE PARSE PARSES THE QUERY AND RETURNS TO THE MAIN * PAR00050
CCC    *  PROGRAM THE CODE GENERETED BY PARSING THE QUERY.             * PAR00060
CCC    *  CODE IS RETURNED TO THE ARRAY CODE AND THE NUMERICAL CODE IS  * PAR00070
CCC    *  EXPLAINT AT THE SUBROUTINES WHERE IT IS CREATED.             * PAR00080
CCC    *                                                               * PAR00090
CCC    ***************************************************************** PAR00100
CCC                                                                      PAR00110
CC                                                                       PAR00120
       SUBROUTINE PARSE(QRY,CODE,MASK,ARG2,ARG4,ICCL1,IER)               PAR00130
       INTEGER*2 QRY(80),MASK(8),CODE(20)                                PAR00140
       COMMON /CHAR/A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,         PAR00150
      -BLANK,COMMA,STAR,LPAR,RPAR                                        PAR00160
       INTEGER*2 A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,            PAR00170
      -BLANK,COMMA,STAR,LPAR,RPAR                                        PAR00180
       REAL ARG2,ARG4                                                    PAR00190
       IER=0                                                             PAR00200
```

```
C                                                                       PAR00210
C   IF  COMMENTS   IGNORE THE  QUERY                                    PAR00220
C                                                                       PAR00230
      IF(QRY(1).NE.STAR) GO TO 10                                       PAR00240
      CODE(1)=10                                                        PAR00250
      RETURN                                                            PAR00260
10    DO 1 J=1,80                                                       PAR00270
      IF(QRY(J).EQ.BLANK) GO TO 1                                       PAR00280
      ICOL1=J                                                           PAR00290
      GO TO 20                                                          PAR00300
1     CONTINUE                                                          PAR00310
      IER=1                                                             PAR00320
      RETURN                                                            PAR00330
C                                                                       PAR00340
C   PROCESS  THE  TYPE  FIELD  OF  THE  COMMAND                         PAR00350
C                                                                       PAR00360
20    CALL TYPE(QRY,ICOL1,CODE,IER)                                     PAR00370
      IF(IER.NE.0) RETURN                                               PAR00380
C                                                                       PAR00390
C   IF  CONTROL  TYPE  COMMAND  RETURN  OTHERWISE  EXAMINE              PAR00400
C   THE REMAINING FIELDS.                                               PAR00410
C                                                                       PAR00420
      IF(CODE(1).LT.10) RETURN                                          PAR00430
      CALL GETNCH(QRY,ICOL1,BLANK,IER)                                  PAR00440
      IF(IER.NE.0) RETURN                                               PAR00450
      CALL QUAL(QRY,ICOL1,CODE,IER)                                     PAR00460
      IF(IER.NE.0) RETURN                                               PAR00470
      CALL GETNCH(QRY,ICOL1,BLANK,IER)                                  PAR00480
      IF(IER.NE.0.AND.CODE(1).NE.17.AND.CODE(2).NE.1) RETURN            PAR00490
      IF(IER.EQ.0) GO TO 21                                             PAR00500
      IER=0                                                             PAR00510
      RETURN                                                            PAR00520
21    CALL MASKA(QRY,ICOL1,CODE,MASK,STAR,BLANK,IER)                    PAR00530
      IF(IER.NE.0) RETURN                                               PAR00540
      CALL GETNCH(QRY,ICOL1,BLANK,IER)                                  PAR00550
      IF(IER.EQ.0) GO TO 30                                             PAR00560
      IF(IER.EQ.201) IER=0                                              PAR00570
      RETURN                                                            PAR00580
30    CALL COND(QRY,ICOL1,CODE,ARG2,ARG4,IER)                           PAR00590
      IF(IER.EQ.501) GO TO 40                                           PAR00600
      IF(IER.NE.0) RETURN                                               PAR00610
      CALL GETNCH(QRY,ICOL1,BLANK,IER)                                  PAR00620
      IF(IER.EQ.0) GO TO 40                                             PAR00630
      IF(IER.EQ.201) IER=0                                              PAR00640
      RETURN                                                            PAR00650
40    CALL PRINT(QRY,ICOL1,CODE,IER)                                    PAR00660
      RETURN                                                            PAR00670
      END                                                               PAR00680
```

```
C                                                                       PAR00690
C                                                                       PAR00700
C  ***************************************************************** PAR00710
C  *                                                               * PAR00720
C  *      SUBROUTINE TYPE PARSES THE TYPE FIELD OF THE COMMAND     * PAR00730
C  *                  INTERNAL    CODES                            * PAR00740
C  *                  ---------------------                        * PAR00750
C  *            V-ERIFY                :  1                       * PAR00760
C  *            NOV-ERIFY              :  2                       * PAR00770
C  *            P-ROMPT                :  3                       * PAR00780
C  *            NOP-ROMPT              :  4                       * PAR00790
C  *            H-EADING               :  5                       * PAR00800
C  *            E-ND                   :  6                       * PAR00810
C  *            SET                    :  7                       * PAR00820
C  *                                                               * PAR00830
C  *            D-ISPLAY               :  11                      * PAR00840
C  *            COUNT                  :  12                      * PAR00850
C  *            A-DD                   :  13                      * PAR00860
C  *            AV-ERAGE               :  14                      * PAR00870
C  *             -ORT                  :  15                      * PAR00880
C  *            AC-TIVATE              :  16                      * PAR00890
C  *            DE-ACTIVATE            :  17                      * PAR00900
C  *                                                               * PAR00910
C  ***************************************************************** PAR00920
C                                                                       PAR00930
C                                                                       PAR00940
      SUBROUTINE TYPE(QRY,ICOL1,CODE,IER)                               PAR00950
      INTEGER*2 QRY(80),CODE(20)                                        PAR00960
      COMMON /CHAR/A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,         PAR00970
     -BLANK,COMMA,STAR,LPAR,RPAR                                        PAR00980
      INTEGER*2 A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,            PAR00990
     -BLANK,COMMA,STAR,LPAR,RPAR                                        PAR01000
      IER=0                                                             PAR01010
      IF(QRY(ICOL1).NE.D) GO TO 10                                      PAR01020
      CODE(1)=11                                                        PAR01030
      IF(QRY(ICOL1+1).EQ.E) CODE(1)=17                                  PAR01040
      RETURN                                                            PAR01050
10    IF(QRY(ICOL1).NE.C) GO TO 20                                      PAR01060
      CODE(1)=12                                                        PAR01070
      RETURN                                                            PAR01080
20    IF(QRY(ICOL1).NE.A) GO TO 30                                      PAR01090
      IF(QRY(ICOL1+1).NE.V) GO TO 25                                    PAR01100
      CODE(1)=14                                                        PAR01110
      RETURN                                                            PAR01120
25    CODE(1)=13                                                        PAR01130
      IF(QRY(ICOL1+1).EQ.D.OR.QRY(ICOL1+1).EQ.BLANK) RETURN             PAR01140
      CODE(1)=16                                                        PAR01150
      IF(QRY(ICOL1+1).EQ.C) RETURN                                      PAR01160
```

```
      IER=101                                                          PAR01170
      RETURN                                                           PAR01180
30    IF(QRY(ICOL1).NE.S) GO TO 40                                     PAR01190
      CODE(1)=15                                                       PAR01200
      IF(QRY(ICOL1+1).EQ.0.OR.QRY(ICOL1+1).EQ.BLANK) RETURN            PAR01210
      CODE(1)=7                                                        PAR01220
      IF(QRY(ICOL1+1).EQ.E.AND.QRY(ICOL1+2).EQ.T) RETURN               PAR01230
      IER=102                                                          PAR01240
      RETURN                                                           PAR01250
40    IF(QRY(ICOL1).NE.V) GO TO 50                                     PAR01260
      CODE(1)=1                                                        PAR01270
      RETURN                                                           PAR01280
50    IF(QRY(ICOL1).NE.P) GO TO 60                                     PAR01290
      CODE(1)=3                                                        PAR01300
      RETURN                                                           PAR01310
60    IF(QRY(ICOL1).NE.H) GO TO 70                                     PAR01320
      CODE(1)=5                                                        PAR01330
      RETURN                                                           PAR01340
70    IF(QRY(ICOL1).NE.N.AND.QRY(ICOL1+1).NE.0) GO TO 80               PAR01350
      CODE(1)=2                                                        PAR01360
      IF(QRY(ICOL1+2).EQ.V) RETURN                                     PAR01370
      CODE(1)=4                                                        PAR01380
      IF(QRY(ICOL1+2).EQ.P) RETURN                                     PAR01390
      IER=103                                                          PAR01400
      RETURN                                                           PAR01410
80    CODE(1)=6                                                        PAR01420
      IF(QRY(ICOL1).EQ.E) RETURN                                       PAR01430
      IER=104                                                          PAR01440
      RETURN                                                           PAR01450
      END                                                              PAR01460
C                                                                      PAR01470
CCC                                                                    PAR01480
CCC   ************************************************************     PAR01490
CCC   *                                                          *     PAR01500
CCC   *      SUBROUTINE  QUAL  PARSES THE QUALIFIED FIELD        *     PAR01510
CCC   *                    INTERNAL    CODES                     *     PAR01520
CCC   *                  ----------------------                  *     PAR01530
CCC   *                  A-LL              :  1                  *     PAR01540
CCC   *                  C-OLUMNS          :  2                  *     PAR01550
CCC   *                  R-OWS             :  3                  *     PAR01560
CCC   *                                                          *     PAR01570
CCC   ************************************************************     PAR01580
CCC                                                                    PAR01590
C                                                                      PAR01600
      SUBROUTINE QUAL(QRY,ICOL1,CODE,IER)                              PAR01610
      INTEGER*2 QRY(80),CODE(20)                                       PAR01620
      COMMON /CHAR/A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,        PAR01630
     -BLANK,COMMA,STAR,LPAR,RPAR                                       PAR01640
```

```
      INTEGER*2 A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,          PAR01650
     -BLANK,COMMA,STAR,LPAR,RPAR                                     PAR01660
      IER=0                                                          PAR01670
      IF(QRY(ICOL1).NE.A) GO TO 10                                   PAR01680
      CODE(2)=1                                                      PAR01690
      RETURN                                                         PAR01700
10    IF(QRY(ICOL1).NE.C) GO TO 20                                   PAR01710
      CODE(2)=2                                                      PAR01720
      RETURN                                                         PAR01730
20    IF(QRY(ICOL1).NE.R) IER=301                                    PAR01740
      CODE(2)=3                                                      PAR01750
      RETURN                                                         PAR01760
      END                                                            PAR01770
C                                                                    PAR01780
CC                                                                   PAR01790
CCC  ***************************************************************** PAR01800
CCC  *                                                               * PAR01810
CCC  *      SUBROUTINE MASKA PROCESS THE MASK FIELD OF THE COMMAND   * PAR01820
CCC  *              INTERNAL              CODE                       * PAR01830
CCC  *           ------------------------------                      * PAR01840
CCC  *             MASK ALL *              :  1                      * PAR01850
CCC  *             MASK NOT ALL *          :  2                      * PAR01860
CCC  *             MASK WITHOUT ANY *      :  3                      * PAR01870
CCC  *                                                               * PAR01880
CCC  ***************************************************************** PAR01890
CCC                                                                  PAR01900
C                                                                    PAR01910
      SUBROUTINE MASKA(QRY,ICOL1,CODE,MASK,STAR,BLANK,IER)           PAR01920
      INTEGER*2 QRY(1),CODE(1),MASK(1),STAR,BLANK                    PAR01930
      IFLAG1=0                                                       PAR01940
      IFLAG2=0                                                       PAR01950
      DO 1 I=1,8                                                     PAR01960
      MASK(I)=STAR                                                   PAR01970
      IF(IFLAG1.NE.0) GO TO 1                                        PAR01980
      IF(QRY(ICOL1+I-1).NE.BLANK) GO TO 10                           PAR01990
      IFLAG1=1                                                       PAR02000
      GO TO 1                                                        PAR02010
10    IF(QRY(ICOL1+I-1).NE.STAR) IFLAG2=IFLAG2+1                     PAR02020
      MASK(I)=QRY(ICOL1+I-1)                                         PAR02030
1     CONTINUE                                                       PAR02040
      CODE(3)=2                                                      PAR02050
      IF(IFLAG2.EQ.0) CODE(3)=1                                      PAR02060
      IF(IFLAG2.EQ.8) CODE(3)=3                                      PAR02070
      RETURN                                                         PAR02080
      END                                                            PAR02090
C                                                                    PAR02100
CC                                                                   PAR02110
C    ***************************************************************** PAR02120
```

```
C    *                                                                    * PAR02130
C    *      SUBROUTINE  COND  PARSES THE CONDITION FIELD OF THE           * PAR02140
C    *   QUERY AND GENERATES THE CORRESPONDING CODE.                      * PAR02150
C    *   THERE ARE 3 KINDS OF SIMPLE CONDITIONAL PHRASES :                * PAR02160
C    *      1.  <ARG1> <RELOP> <ARG2>                                     * PAR02170
C    *            INTERNAL    CODES                                       * PAR02180
C    *            ------------------                                      * PAR02190
C    *          ARG1                                  RELOP             * PAR02200
C    *         ------                                 -------           * PAR02210
C    *    X             :1                          EQ     : 1          * PAR02220
C    *    S, OR C       :2                          NE     : 2          * PAR02230
C    *    L             :3                          GT     : 3          * PAR02240
C    *    U             :4                          GE     : 4          * PAR02250
C    *    D             :5                          LT     : 5          * PAR02260
C    *    P             :6                          LE     : 6          * PAR02270
C    *    W             :7                                              * PAR02280
C    *          ARG2                                                    * PAR02290
C    *         ------                                                   * PAR02300
C    *        A.  AS ARG1                                               * PAR02310
C    *        B.  ANY REAL OR INTEGER NUMBER                            * PAR02320
C    *        2.  <ARG1> <MIN> OR <ARG1> <MAX>                          * PAR02330
C    *            INTERNAL    CODES                                     * PAR02340
C    *            ------------------                                    * PAR02350
C    *            MIN     : 7                                           * PAR02360
C    *            MAX     : 8                                           * PAR02370
C    *        3.  STATUS <ARG> OR ST <ARG>                              * PAR02380
C    *            INTERNAL    CODES                                     * PAR02390
C    *            ------------------                                    * PAR02400
C    *                 STATUS OR ST  : 9                                * PAR02410
C    *                      ARG                                         * PAR02420
C    *                     -----                                        * PAR02430
C    *     IN : 00         FC : 07         TR : 14                      * PAR02440
C    *     BS : 01         BC : 08         PP : 15                      * PAR02450
C    *     LL : 02         VR : 09         IE : 16                      * PAR02460
C    *     UL : 03         SR : 10         AE : 17                      * PAR02470
C    *     EQ : 04         RR : 11         CE : 18                      * PAR02480
C    *     VC : 05         FR : 12         DE : 19                      * PAR02490
C    *     SC : 06         ER : 13                                      * PAR02500
C    *      THERE ARE 2 KINDS OF COMPOUNT CONDITIONAL PHRASES :         * PAR02510
C    *      1. <1ST SIMP> <RELOP> <1ST SIMP>                            * PAR02520
C    *      2. <1ST SIMP> <RELOP> <2ND SIMP>                            * PAR02530
C    *            INTERNAL    CODES                                     * PAR02540
C    *            ------------------                                    * PAR02550
C    *             OR     : 1                                           * PAR02560
C    *             AND    : 2                                           * PAR02570
C    *   FOR MORE DETAILS SEE USER'S MANUAL                             * PAR02580
C    *                                                                  * PAR02590
C    ********************************************************************** PAR02600
```

```
      SUBROUTINE COND(QRY,ICOL1,CODE,ARG2,ARG4,IER)                    PAR02610
      INTEGER*2 QRY(80),CODE(20)                                       PAR02620
      COMMON /CHAR/A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,        PAR02630
     -BLANK,COMMA,STAR,LPAR,RPAR                                       PAR02640
      INTEGER*2 A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,           PAR02650
     -BLANK,COMMA,STAR,LPAR,RPAR                                       PAR02660
      REAL ARG2,ARG4                                                   PAR02670
      IER=0                                                            PAR02680
C                                                                      PAR02690
C                                                                      PAR02700
C     CHECK IF WORD 'FOR' EXIST                                        PAR02710
C                                                                      PAR02720
      IF(QRY(ICOL1).EQ.F) GO TO 10                                     PAR02730
      IER=501                                                          PAR02740
      RETURN                                                           PAR02750
10    IF(QRY(ICOL1+1).EQ.O.AND.QRY(ICOL1+2).EQ.R) GO TO 20             PAR02760
      IER=502                                                          PAR02770
      RETURN                                                           PAR02780
20    ICOL1=ICOL1+3                                                    PAR02790
      IF(QRY(ICOL1).EQ.BLANK) CALL GETNCH(QRY,ICOL1,BLANK,IER)         PAR02800
      IF(IER.NE.0) RETURN                                              PAR02810
C                                                                      PAR02820
C                                                                      PAR02830
C     CHECK FOR LEFT PARENTHESIS                                       PAR02840
      IF(QRY(ICOL1).EQ.LPAR) GO TO 30                                  PAR02850
      IER=503                                                          PAR02860
      RETURN                                                           PAR02870
30    ICOL1=ICOL1+1                                                    PAR02880
      IF(QRY(ICOL1).EQ.BLANK) CALL GETNCH(QRY,ICOL1,BLANK,IER)         PAR02890
      IF(IER.NE.0) RETURN                                              PAR02900
C                                                                      PAR02910
C                                                                      PAR02920
C     IF 1ST KIND OF CONDITIONAL PHRASE CALL SIMCOD                    PAR02930
C                                                                      PAR02940
      INDEX=4                                                          PAR02950
      IF(QRY(ICOL1).EQ.S.AND.QRY(ICOL1+1).EQ.T) GO TO 40               PAR02960
      CALL SIMCOD(QRY,ICOL1,CODE,ARG2,INDEX,IER)                       PAR02970
      IF(IER.NE.0)RETURN                                               PAR02980
      DO 1 J=1,15                                                      PAR02990
      IF(QRY(ICOL1+J).NE.BLANK) GO TO 1                                PAR03000
      ICOL1=ICOL1+J-1                                                  PAR03010
      GO TO 50                                                         PAF03020
1     CONTINUE                                                         PAR03030
      IER=504                                                          PAR03040
      RETURN                                                           PAR03050
50    IF(QRY(ICOL1).EQ.RPAR) RETURN                                    PAR03060
      CALL GETNCH(QRY,ICOL1,BLANK,IER)                                 PAR03070
      IF(IER.NE.0) RETURN                                              PAR03080
```

```
      IF(QRY(ICOL1).EQ.RPAR) RETURN                                    PAR03090
      IER=505                                                          PAR03100
      IF(CODE(5).EQ.7.OR.CODE(5).EQ.8) RETURN                          PAR03110
      IER=0                                                            PAR03120
C                                                                      PAR03130
CC                                                                     PAR03140
C     FIND WHAT RELOP (OR , AND) EXIST                                 PAR03150
C                                                                      PAR03160
      IF(QRY(ICOL1).NE.O) GO TO 60                                     PAR03170
      CODE(7)=1                                                        PAR03180
      IF(QRY(ICOL1+1).EQ.R) GO TO 61                                   PAR03190
      IER=506                                                          PAR03200
      RETURN                                                           PAR03210
60    IF(QRY(ICOL1).NE.A) GO TO 70                                     PAR03220
      CODE(7)=2                                                        PAR03230
      IF(QRY(ICOL1+1).EQ.N.AND.QRY(ICOL1+2).EQ.D) GO TO 61             PAR03240
      IER=507                                                          PAR03250
      RETURN                                                           PAR03260
70    IER=508                                                          PAR03270
      RETURN                                                           PAR03280
61    CALL GETNCH(QRY,ICOL1,BLANK,IER)                                 PAR03290
      IF(IER.NE.O) RETURN                                              PAR03300
      INDEX=8                                                          PAR03310
C                                                                      PAR03320
CC                                                                     PAR03330
C     IF COMPOUNT COMDITION CALL SIMCOD                                PAR03340
C                                                                      PAR03350
      IF(QRY(ICOL1).EQ.S.AND.QRY(ICOL1+1).EQ.T) GO TO 40               PAR03360
      CALL SIMCOD(QRY,ICOL1,CODE,ARG4,INDEX,IER)                       PAR03370
      IF(IER.NE.O) RETURN                                              PAR03380
      GO TO 301                                                        PAR03390
C                                                                      PAR03400
CC                                                                     PAR03410
C     PARSE THE 2ND KIND OF CONDITION. FIND THE OPERAND OF             PAR03420
C     STATUS                                                           PAR03430
C                                                                      PAR03440
40    CODE(INDEX)=9                                                    PAR03450
      CALL GETNCH(QRY,ICOL1,BLANK,IER)                                 PAR03460
      IF(IER.NE.O) RETURN                                              PAR03470
      IF(QRY(ICOL1).NE.B) GO TO 110                                    PAR03480
      CODE(INDEX+1)=1                                                  PAR03490
      IF(QRY(ICOL1+1).EQ.S) GO TO 301                                  PAR03500
      IF (QRY(ICOL1+1).EQ.C) GO TO 143                                 PAR03510
      IER=511                                                          PAR03520
      RETURN                                                           PAR03530
110   IF(QRY(ICOL1).NE.L) GO TO 120                                    PAR03540
      CODE(INDEX+1)=2                                                  PAR03550
      IF(QRY(ICOL1+1).EQ.L) GO TO 301                                  PAR03560
```

```
      IER=512                                                          PAR03570
      RETURN                                                           PAR03580
120   IF(QRY(ICOL1).NE.U) GO TO 130                                    PAR03590
      CODE(INDEX+1)=3                                                  PAR03600
      IF(QRY(ICOL1+1).EQ.L) GO TO 301                                  PAR03610
      IER=513                                                          PAR03620
      RETURN                                                           PAR03630
130   IF(QRY(ICOL1).NE.E) GO TO 140                                    PAR03640
      CODE(INDEX+1)=4                                                  PAR03650
      IF(QRY(ICOL1+1).EQ.Q) GO TO 301                                  PAR03660
      IF(QRY(ICOL1+1).EQ.R) GO TO 131                                  PAR03670
      IER=514                                                          PAR03680
      RETURN                                                           PAR03690
140   IF(QRY(ICOL1+1).NE.C) GO TO 150                                  PAR03700
      IF(QRY(ICOL1).NE.V) GO TO 141                                    PAR03710
      CODE(INDEX+1)=5                                                  PAR03720
      GO TO 301                                                        PAR03730
141   IF(QRY(ICOL1).NE.S) GO TO 142                                    PAR03740
      CODE(INDEX+1)=6                                                  PAR03750
      GO TO 301                                                        PAR03760
142   IF(QRY(ICOL1).NE.F) GO TO 143                                    PAR03770
      CODE(INDEX+1)=7                                                  PAR03780
      GO TO 301                                                        PAR03790
143   CODE(INDEX+1)=8                                                  PAR03800
      IF(QRY(ICOL1).EQ.B) GO TO 301                                    PAR03810
      IER=515                                                          PAR03820
      RETURN                                                           PAR03830
150   IF(QRY(ICOL1+1).NE.R) GO TO 160                                  PAR03840
      IF(QRY(ICOL1).NE.V) GO TO 151                                    PAR03850
      CODE(INDEX+1)=9                                                  PAR03860
      GO TO 301                                                        PAR03870
151   IF(QRY(ICOL1).NE.S) GO TO 152                                    PAR03880
      CODE(INDEX+1)=10                                                 PAR03890
      GO TO 301                                                        PAR03900
152   IF(QRY(ICOL1).NE.R) GO TO 153                                    PAR03910
      CODE(INDEX+1)=11                                                 PAR03920
      GO TO 301                                                        PAR03930
153   IF(QRY(ICOL1).NE.F) GO TO 154                                    PAR03940
      CODE(INDEX+1)=12                                                 PAR03950
      GO TO 301                                                        PAR03960
154   IF(QRY(ICOL1).NE.E) GO TO 155                                    PAR03970
131   CODE(INDEX+1)=13                                                 PAR03980
      GO TO 301                                                        PAR03990
155   CODE(INDEX+1)=14                                                 PAR04000
      IF(QRY(ICOL1).EQ.T) GO TO 301                                    PAR04010
      IER=516                                                          PAR04020
      RETURN                                                           PAR04030
160   IF(QRY(ICOL1+1).NE.P) GO TO 170                                  PAR04040
```

```
      CODE(INDEX+1)=15                                                  PAR04050
      IF(QRY(ICOL1).EQ.P) GO TO 301                                     PAR04060
      IER=517                                                           PAR04070
      RETURN                                                            PAR04080
170   IF(QRY(ICOL1+1).NE.E) GO TO 180                                   PAR04090
      IF(QRY(ICOL1).NE.I) GO TO 171                                     PAR04100
      CODE(INDEX+1)=16                                                  PAR04110
      GO TO 301                                                         PAR04120
171   IF(QRY(ICOL1).NE.A) GO TO 172                                     PAR04130
      CODE(INDEX+1)=17                                                  PAR04140
      GO TO 301                                                         PAR04150
172   IF(QRY(ICOL1).NE.C) GO TO 173                                     PAR04160
      CODE(INDEX+1)=18                                                  PAR04170
      GO TO 301                                                         PAR04180
173   CODE(INDEX+1)=19                                                  PAR04190
      IF(QRY(ICOL1).EQ.D) GO TO 301                                     PAR04200
      IER=518                                                           PAR04210
      RETURN                                                            PAR04220
301   DO 2 J=1,15                                                       PAR04230
      IF(QRY(ICOL1+J).NE.BLANK) GO TO 2                                 PAR04240
      ICOL1=ICOL1+J-1                                                   PAR04250
      GO TO 80                                                          PAR04260
2     CONTINUE                                                          PAR04270
      IER=519                                                           PAR04280
      RETURN                                                            PAR04290
C                                                                       PAR04300
CC                                                                      PAR04310
C     CHECK FOR RIGHT PARENTHESIS                                       PAR04320
C                                                                       PAR04330
80    IF(QRY(ICOL1).EQ.RPAR) RETURN                                     PAR04340
      CALL GETNCH(QRY,ICOL1,BLANK,IER)                                  PAR04350
      IF(IER.NE.0) RETURN                                               PAR04360
      IF(QRY(ICOL1).EQ.RPAR) RETURN                                     PAR04370
180   IF(QRY(ICOL1).EQ.I.AND.QRY(ICOL1+1).EQ.N) RETURN                  PAR04380
      IER=520                                                           PAR04390
      RETURN                                                            PAR04400
      END                                                               PAR04410
C                                                                       PAR04420
CC                                                                      PAR04430
CCC                                                                     PAR04440
CCCC                                                                    PAR04450
CCCCC                                                                   PAR04460
C     ********************************************************************PAR04470
C     *                                                                 * PAR04480
C     *     SUBROUTINE  SIMCOD  PARSES AND GENERATES THE                * PAR04490
C     *  CORRESPONDING CODE FOR SIMPLE CONDITIONAL PHRASES 1ST          * PAR04500
C     *  AND 2ND KIND.                                                  * PAR04510
C     *                                                                 * PAR04520
```

```
C    ********************************************************************** PAR04530
       SUBROUTINE SIMCOD(QRY,ICOL1,CODE,ARG2,INDEX,IER)                     PAR04540
       INTEGER*2 QRY(80),CODE(20)                                           PAR04550
       COMMON /CHAR/A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,            PAR04560
      -BLANK,COMMA,STAR,LPAR,RPAR                                           PAR04570
       INTEGER*2 A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,               PAR04580
      -BLANK,COMMA,STAR,LPAR,RPAR                                           PAR04590
       REAL ARG2,ARG4                                                       PAR04600
       IER=0                                                                PAR04610
       IFLAG=0                                                              PAR04620
C                                                                           PAR04630
C    FIND ARG1 AND ARG2.                                                    PAR04640
C                                                                           PAR04650
99     IF(QRY(ICOL1).NE.X) GO TO 20                                         PAR04660
       CODE(INDEX+IFLAG)=1                                                  PAR04670
       GO TO 100                                                            PAR04680
20     IF(QRY(ICOL1).NE.S.AND.QRY(ICOL1).NE.C) GO TO 30                     PAR04690
       CODE(INDEX+IFLAG)=2                                                  PAR04700
       GO TO 100                                                            PAR04710
30     IF(QRY(ICOL1).NE.L) GO TO 40                                         PAR04720
       CODE(INDEX+IFLAG)=3                                                  PAR04730
       GO TO 100                                                            PAR04740
40     IF(QRY(ICOL1).NE.U) GO TO 50                                         PAR04750
       CODE(INDEX+IFLAG)=4                                                  PAR04760
       GO TO 100                                                            PAR04770
50     IF(QRY(ICOL1).NE.D) GO TO 60                                         PAR04780
       CODE(INDEX+IFLAG)=5                                                  PAR04790
       GO TO 100                                                            PAR04800
60     IF(QRY(ICOL1).NE.P) GO TO 70                                         PAR04810
       CODE(INDEX+IFLAG)=6                                                  PAR04820
       GO TO 100                                                            PAR04830
70     IF(QRY(ICOL1).NE.W) GO TO 80                                         PAR04840
       CODE(INDEX+IFLAG)=7                                                  PAR04850
       GO TO 100                                                            PAR04860
80     IF(IFLAG.NE.0) GO TO 90                                              PAR04870
       IER=601                                                              PAR04880
       RETURN                                                               PAR04890
90     DO 1 J=1,12                                                          PAR04900
       IF(QRY(ICOL1+J).NE.BLANK) GO TO 1                                    PAR04910
       ICOL2=ICOL1+J-1                                                      PAR04920
       GO TO 91                                                             PAR04930
1      CONTINUE                                                             PAR04940
       IER=602                                                              PAR04950
       RETURN                                                               PAR04960
91     IF(QRY(ICOL2).EQ.RPAR) ICOL2=ICOL2-1                                 PAR04970
       ARG2=VAL(QRY,ICOL1,ICOL2)                                            PAR04980
       CODE(INDEX+2)=9                                                      PAR04990
       RETURN                                                               PAR05000
```

```
C                                                                       PAR05010
C    FIND RELATIONAL OPERATOR (GT, EQ, E.T.C.)                          PAR05020
100    IF(IFLAG.NE.0) RETURN                                            PAR05030
       IFLAG=2                                                          PAR05040
       CALL GETNCH(QRY,ICOL1,BLANK,IER)                                 PAR05050
       IF(IER.NE.0) RETURN                                              PAR05060
       IF(QRY(ICOL1).NE.G) GO TO 11                                     PAR05070
       IF(QRY(ICOL1+1).NE.T) GO TO 12                                   PAR05080
       CODE(INDEX+1)=3                                                  PAR05090
       GO TO 101                                                        PAR05100
12     CODE(INDEX+1)=4                                                  PAR05110
       IF(QRY(ICOL1+1).EQ.E) GO TO 101                                  PAR05120
       IER=603                                                          PAR05130
       RETURN                                                           PAR05140
11     IF(QRY(ICOL1).NE.L) GO TO 21                                     PAR05150
       IF(QRY(ICOL1+1).NE.T) GO TO 22                                   PAR05160
       CODE(INDEX+1)=5                                                  PAR05170
       GO TO 101                                                        PAR05180
22     CODE(INDEX+1)=6                                                  PAR05190
       IF(QRY(ICOL1+1).EQ.E) GO TO 101                                  PAR05200
       IER=604                                                          PAR05210
       RETURN                                                           PAR05220
21     IF(QRY(ICOL1).NE.E) GO TO 31                                     PAR05230
       CODE(INDEX+1)=1                                                  PAR05240
       IF(QRY(ICOL1+1).EQ.Q) GO TO 101                                  PAR05250
       IER=605                                                          PAR05260
       RETURN                                                           PAR05270
31     IF(QRY(ICOL1).NE.N) GO TO 41                                     PAR05280
       CODE(INDEX+1)=2                                                  PAR05290
       IF(QRY(ICOL1+1).EQ.E) GO TO 101                                  PAR05300
       IER=606                                                          PAR05310
       RETURN                                                           PAR05320
41     IF(QRY(ICOL1).NE.M) GO TO 51                                     PAR05330
       IF(QRY(ICOL1+1).NE.I) GO TO 52                                   PAR05340
       CODE(INDEX+1)=7                                                  PAR05350
       IF(QRY(ICOL1+2).EQ.N) RETURN                                     PAR05360
       IER=607                                                          PAR05370
       RETURN                                                           PAR05380
52     CODE(INDEX+1)=8                                                  PAR05390
       IF(QRY(ICOL1+1).EQ.A.AND.QRY(ICOL1+2).EQ.X) RETURN               PAR05400
       IER=608                                                          PAR05410
       RETURN                                                           PAR05420
51     IER=609                                                          PAR05430
       RETURN                                                           PAR05440
101    CALL GETNCH(QRY,ICOL1,BLANK,IER)                                 PAR05450
       IF(IER.EQ.0) GO TO 99                                            PAR05460
       RETURN                                                           PAR05470
       END                                                              PAR05480
```

```
C                                                                       PAR05490
CC    ********************************************************************* PAR05500
CC    *                                                                   * PAR05510
CC    *      SUBROUTINE  PRINT  PARSES AND GENERATES CODE FOR THE         * PAR05520
CC    *   PRINT FIELD OF THE QUERY.                                       * PAR05530
CC    *   AS INTERNAL CODES THE NUMBERS 1-7 ARE USED AND 1 CORRESPONDS    * PAR05540
CC    *   TO THE FIRST ELEMENT APPEARING AT THE PRINT FIELD 2 TO THE      * PAR05550
CC    *   SECOND...E.T.C.                                                 * PAR05560
CC    *   E.G.                                                            * PAR05570
CC    *   FOR THE PRINT FIELD 'D X C' THE SEQUENCE                        * PAR05580
CC    *            2 3 0 0 1 0 0                                          * PAR05590
CC    *   WILL BE GENERATED.                                              * PAR05600
CC    *                                                                   * PAR05610
C     ********************************************************************* PAR05620
      SUBROUTINE PRINT(QRY,ICOL1,CODE,IER)                              PAR05630
      INTEGER*2 QRY(80),CODE(20)                                        PAR05640
      COMMON /CHAR/A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,         PAR05650
     -BLANK,COMMA,STAR,LPAR,RPAR                                        PAR05660
      INTEGER*2 A,B,C,D,E,F,G,H,I,L,M,N,O,P,Q,R,S,T,U,V,W,X,            PAR05670
     -BLANK,COMMA,STAR,LPAR,RPAR                                        PAR05680
      IER=0                                                             PAR05690
      MET=0                                                             PAR05700
      CODE(11)=1                                                        PAR05710
      DO 1 J=ICOL1,80                                                   PAR05720
      IF(QRY(J).EQ.BLANK.OR.QRY(J).EQ.COMMA) GO TO 1                    PAR05730
      MET=MET+1                                                         PAR05740
      IF(QRY(J).NE.X) GO TO 10                                          PAR05750
      CODE(12)=MET                                                      PAR05760
      GO TO 1                                                           PAR05770
10    IF(QRY(J).NE.L) GO TO 20                                          PAR05780
      CODE(14)=MET                                                      PAR05790
      GO TO 1                                                           PAR05800
20    IF(QRY(J).NE.U) GO TO 30                                          PAR05810
      CODE(15)=MET                                                      PAR05820
      GO TO 1                                                           PAR05830
30    IF(QRY(J).NE.D) GO TO 40                                          PAR05840
      CODE(16)=MET                                                      PAR05850
      GO TO 1                                                           PAR05860
40    IF(QRY(J).NE.P) GO TO 50                                          PAR05870
      CODE(17)=MET                                                      PAR05880
      GO TO 1                                                           PAR05890
50    IF(QRY(J).NE.W) GO TO 60                                          PAR05900
      CODE(18)=MET                                                      PAR05910
      GO TO 1                                                           PAR05920
60    CODE(13)=MET                                                      PAR05930
      IF(QRY(J).EQ.C.OR.QRY(J).EQ.S) GO TO 1                            PAR05940
      GO TO 70                                                          PAR05950
1     CONTINUE                                                          PAR05960
```

```
      RETURN                                                            PAR05970
70    IER=701                                                           PAR05980
      RETURN                                                            PAR05990
      END                                                               PAR06000
C                                                                       PAR06010
C     ****************************************************************  PAR06020
C     *                                                              *  PAR06030
C     *      SUBROUTINE  GETNCH  POTITIONS THE POINTER ICOL1 TO THE  *  PAR06040
C     *   FIRST LETTER OF THE FIELD RIGHT AFTER IT CURRENTLY IS.     *  PAR06050
C     *                                                              *  PAR06060
C     ****************************************************************  PAR06070
      SUBROUTINE GETNCH(QRY,ICOL1,BLANK,IER)                            PAR06080
      INTEGER*2 QRY(1),BLANK                                            PAR06090
      IER=0                                                             PAR06100
      ITEMP=ICOL1                                                       PAR06110
      DO 1 I=ITEMP,80                                                   PAR06120
      IF(QRY(I).NE.BLANK) GO TO 1                                       PAR06130
      DO 2  J=I,80                                                      PAR06140
      IF(QRY(J).EQ.BLANK) GO TO 2                                       PAR06150
      ICOL1=J                                                           PAR06160
C     WRITE(6,100) (QRY(K),K=1,ICOL1)                                   PAR06170
100   FORMAT(2X,80A1)                                                   PAR06180
      RETURN                                                            PAR06190
2     CONTINUE                                                          PAR06200
      IER=201                                                           PAR06210
      RETURN                                                            PAR06220
1     CONTINUE                                                          PAR06230
      IER=202                                                           PAR06240
      RETURN                                                            PAR06250
      END                                                               PAR06260
C                                                                       PAR06270
C     ****************************************************************  PAR06280
C     *                                                              *  PAR06290
C     *      FUNCTION  VAL  GETS A CHARACTER STRING IN AN 1-DIMENSION*  PAR06300
C     *   ARRAY X FROM 'IST' LOCATION UP TO 'IED' LOCATION AND RETURNS*  PAR06310
C     *   THE CORRESPONDING VALUE.                                   *  PAR06320
C     *                                                              *  PAR06330
C     ****************************************************************  PAR06340
      REAL FUNCTION VAL*8(X,IST,IED)                                    PAR06350
      INTEGER*2 BLANK,ZERO,MINUS,DECIM,X(80)                            PAR06360
      REAL IFN                                                          PAR06370
      DATA BLANK,ZERO,MINUS,DECIM/' ','0','-','.'/                      PAR06380
      DATA LL/256/                                                      PAR06390
      VAL=0.0D0                                                         PAR06400
      IEXP=0                                                            PAR06410
      SIGN=1.                                                           PAR06420
      IFN=0.0D0                                                         PAR06430
      ITEMP=0                                                           PAR06440
```

```
      MET=0                                                              PAR06450
      DO 1 I=IST,IED                                                     PAR06460
      IF(X(I).EQ.MINUS) GO TO 10                                         PAR06470
      IF(X(I).EQ.BLANK) GO TO 1                                          PAR06480
      MET=MET+1                                                          PAR06490
      IF(X(I).NE.DECIM) GO TO 20                                         PAR06500
      IEXP=MET                                                           PAR06510
      GO TO 1                                                            PAR06520
20    ITEMP=X(I)-ZERO                                                    PAR06530
      ITEMP=ITEMP/LL                                                     PAR06540
C     WRITE(6,100) IFN,ITEMP                                             PAR06550
100   FORMAT(5X,I10,I10)                                                 PAR06560
      IFN=IFN*10+ITEMP                                                   PAR06570
      GO TO 1                                                            PAR06580
10    SIGN=-1.                                                           PAR06590
1     CONTINUE                                                           PAR06600
      LEXP=MET-IEXP                                                      PAR06610
      IF(IEXP.EQ.0) LEXP=0                                               PAR06620
      VAL=SIGN*IFN*10.**(-LEXP)                                          PAR06630
      RETURN                                                             PAR06640
      END                                                                PAR06650
C     ****************************************************************** REA00010
C     *                                                                * REA00020
C     *      SUBROUTINE  READF  ACCEPTS AS INPUT A FILE IN 'STANDARD'  * REA00030
C     *   FORMAT WITH FILE NUMBER 'IFLR' PACKS IT IN SPARSE DATA       * REA00040
C     *   STRUCTURE, WRITES IT IN A DIRECT ACCES FILE 'IFLW' IN PACKED * REA00050
C     *   FORM AND RETURNS THE PACKED FILE IN 'SOLFIL' WITH INFORMATIONS * REA00060
C     *   ABOUT THE SIZE OF FILE :                                     * REA00070
C     *       SOLFIL(1) = SIZE OF FILE IN 4-BYTE WORDS                 * REA00080
C     *       SOLFIL(2) = 0  (DATA STRUCTURE SPARSE)                   * REA00090
C     *       SOLFIL(3) = NUMBER OF ROWS                               * REA00100
C     *       SOLFIL(4) = NUMBER OF COLUMNS                            * REA00110
C     *                                                                * REA00120
C     ****************************************************************** REA00130
C                                                                        REA00140
      SUBROUTINE READF(NROWS,NCOL,IER,IFLR,IFLW)                         REA00150
      REAL DATA(7),SOLIST(1)                                             REA00160
      INTEGER*2 CSTAT(15),SOFLAG(1),STAT                                 REA00170
      LOGICAL*1 BITS(16)                                                 REA00180
      COMMON /INIT/ZERO,TINF,IACT,MEMOR                                  REA00190
      COMMON /SOLPAC/SOLFIL(81920)                                       REA00200
      EQUIVALENCE (SOLFIL(5),SOFLAG(1),SOLIST(1))                        REA00210
      DATA CSTAT/2HIN,2HBS,2HLL,2HUL,2HEQ,2HVC,2HSC,2HFC,2HBC,           REA00220
     -2HVR,2HSR,2HRR,2HFR,2HER,2HTR/                                     REA00230
      IER=0                                                              REA00240
C                                                                        REA00250
C     DEFINE THE TAPE FILE UNDER CP/CMS.                                 REA00260
C     SEE TAPSET SUBROUTINE IN CP/CMS MANUAL.                            REA00270
```

```
C                                                                       REA00280
      CALL TAPSET(2,IFLR,80,17,1,80)                                    REA00290
C                                                                       REA00300
C     READ IN NROWS, NCOL AND IFLP                                      REA00310
C     IF IFLP=0 THERE ARE NOT PENALTIES IN THE FILE                     REA00320
C                                                                       REA00330
      READ(IFLR,100) NROWS,NCOL,IFLGP                                   REA00340
100   FORMAT(I5,30X,I5,10X,I1)                                          REA00350
      SOLFIL(2)=1                                                       REA00360
      SOLFIL(3)=NROWS                                                   REA00370
      SOLFIL(4)=NCOL                                                    REA00380
      NRCOL=NROWS+NCOL                                                  REA00390
      ICFFS=3*NRCOL                                                     REA00400
      INDEX=IOFFS+1                                                     REA00410
      DO 1 I=1,NRCOL                                                    REA00420
      J=I*3-2                                                           REA00430
      DATA(6)=0.                                                        REA00440
      DATA(7)=0.                                                        REA00450
C                                                                       REA00460
C     READ ONE RECORD AT A TIME                                         REA00470
C     IOFFS= THE LAST WORD USED BY NAMES AND FLAGS.                     REA00480
C     INDEX= NEXT AVAILABLE LOCATION TO STORE A NUMBER.                 REA00490
C                                                                       REA00500
      READ(IFLR,200,END=99) SOLIST(J),SOLIST(J+1),STAT,(DATA(L),L=1,5)  REA00510
      IF(IFLGP.NE.0) READ(IFLR,201,END=99) DATA(6),DATA(7)              REA00520
201   FORMAT(E14.5,16X,E14.5)                                           REA00530
      JW=J                                                              REA00540
200   FORMAT(2A4,A2,5E14.5)                                             REA00550
      K=I*6                                                             REA00560
      SCFLAG(K)=INDEX-IOFFS-1                                           REA00570
C                                                                       REA00580
C     FIND THE STATUS OF THE CURRENT RECORD.                            REA00590
      DO 2 J=1,15                                                       REA00600
      IF(STAT.NE.CSTAT(J)) GO TO 2                                      REA00610
      ICOD=J-1                                                          REA00620
      GO TO 90                                                          REA00630
2     CONTINUE                                                          REA00640
      IER=1001                                                          REA00650
      RETURN                                                            REA00660
90    IHELP=ICOD                                                        REA00670
C                                                                       REA00680
C     UPDATE THE 4 FIRST BITS.                                          REA00690
C                                                                       REA00700
      DO 3 J=1,4                                                        REA00710
      BITS(5-J)=.FALSE.                                                 REA00720
      IF(MOD(IHELP,2).EQ.1) BITS(5-J)=.TRUE.                            REA00730
      IHELP=IHELP/2                                                     REA00740
3     CONTINUE                                                          REA00750
```

```
C                                                                       REA00760
C     CHECK IF VALUES ARE ELIGIBLE FOR STORAGE AND                      REA00770
C     UPDATE THE CORRESPNDING BITS.                                     REA00780
C                                                                       REA00790
        BITS(5)=.TRUE.                                                  REA00800
        IF(DATA(1).EQ.0.) GO TO 10                                      REA00810
        BITS(5)=.FALSE.                                                 REA00820
        SCLIST(INDEX)=DATA(1)                                           REA00830
        INDEX=INDEX+1                                                   REA00840
10      BITS(6)=.TRUE.                                                  REA00850
        IF(DATA(2).EQ.0.) GO TO 20                                      REA00860
        BITS(6)=.FALSE.                                                 REA00870
        SCLIST(INDEX)=DATA(2)                                           REA00880
        INDEX=INDEX+1                                                   REA00890
20      BITS(7)=.TRUE.                                                  REA00900
        BITS(10)=.FALSE.                                                REA00910
        IF(DATA(3).EQ.0.) GO TO 30                                      REA00920
        IF(ICOD.EQ.2.OR.ICOD.EQ.4) GO TO 30                             REA00930
        IF(DATA(3).EQ.-TINF) GO TO 25                                   REA00940
        BITS(7)=.FALSE.                                                 REA00950
        SCLIST(INDEX)=DATA(3)                                           REA00960
        INDEX=INDEX+1                                                   REA00970
        GO TO 30                                                        REA00980
25      BITS(10)=.TRUE.                                                 REA00990
30      BITS(8)=.TRUE.                                                  REA01000
        BITS(11)=.FALSE.                                                REA01010
        IF(DATA(4).EQ.0.) GO TO 40                                      REA01020
        IF(ICOD.EQ.3.OR.ICOD.EQ.4) GO TO 40                             REA01030
        IF(DATA(4).EQ.TINF) GO TO 35                                    REA01040
        BITS(8)=.FALSE.                                                 REA01050
        SCLIST(INDEX)=DATA(4)                                           REA01060
        INDEX=INDEX+1                                                   REA01070
        GO TO 40                                                        REA01080
35      BITS(11)=.TRUE.                                                 REA01090
40      BITS(9)=.TRUE.                                                  REA01100
        IF(DATA(5).EQ.0.) GO TO 50                                      REA01110
        BITS(9)=.FALSE.                                                 REA01120
        SCLIST(INDEX)=DATA(5)                                           REA01130
        INDEX=INDEX+1                                                   REA01140
C                                                                       REA01150
C     CHECK THE STATUS OF PENALTIES.                                    REA01160
C                                                                       REA01170
50      IHELP=10                                                        REA01180
        IF(DATA(6).EQ.0.) GO TO 60                                      REA01190
        IHELP=30                                                        REA01200
        IF(DATA(6).EQ.TINF) GO TO 60                                    REA01210
        IHELP=20                                                        REA01220
        SCLIST(INDEX)=DATA(6)                                           REA01230
```

```
      INDEX= INDEX+1                                                    REA01240
60    IHELP1=1                                                          REA01250
      IF(DATA(7).EQ.0.) GO TO 70                                        REA01260
      IHELP1=3                                                          REA01270
      IF(DATA(7).EQ.TINF) GO TO 70                                      REA01280
      IHELP1=2                                                          REA01290
      SOLIST(INDEX)=DATA(7)                                             REA01300
      INDEX=INDEX+1                                                     REA01310
70    IHELP=IHELP+IHELP1                                                REA01320
      ICOD=IHELP-11                                                     REA01330
      IF(IHELP.LE.13) GO TO 80                                          REA01340
      ICOD=IHELP-18                                                     REA01350
      IF(IHELP.LE.23) GO TO 80                                          REA01360
      ICOD=IHELP-25                                                     REA01370
80    DO 4 J=1,4                                                        REA01380
      BITS(16-J)=.FALSE.                                                REA01390
      IF(MOD(ICOD,2).EQ.1) BITS(16-J)=.TRUE.                            REA01400
      ICOD=ICOD/2                                                       REA01410
4     CONTINUE                                                          REA01420
      BITS(16)=.FALSE.                                                  REA01430
      IHELP=0                                                           REA01440
C                                                                       REA01450
C     FIND THE CORRESPONDING DECIMAL NUMBER TO THE                      REA01460
C     BITS PATTERN AND STORE IT.                                        REA01470
C                                                                       REA01480
      DO 5 J=1,16                                                       REA01490
      IF(BITS(17-J)) IHELP=IHELP+2**(J-1)                               REA01500
5     CONTINUE                                                          REA01510
      SOFLAG(K-1)=IHELP                                                 REA01520
1     CONTINUE                                                          REA01530
      INDEX=INDEX+4                                                     REA01540
C                                                                       REA01550
C     GIVE INFORMATIONS FOR THE SIZE OF PACKED FILE.                    REA01560
C                                                                       REA01570
      WRITE(6,300) INDEX                                                REA01580
300   FORMAT(5X,'WORDS REQUIRED FOR PACKED FILE: ',I6)                  REA01590
      SOLFIL(1)=INDEX                                                   REA01600
C                                                                       REA01610
C     WRITE THE PACKED FILE                                             REA01620
C                                                                       REA01630
      WRITE(IFLW'1) (SOLFIL(IJ),IJ=1,INDEX)                             REA01640
      RETURN                                                            REA01650
99    WRITE(6,999) SOLIST(JW),SOLIST(JW+1),STAT,(DATA(L),L=1,5)         REA01660
999   FORMAT(2X,2A4,A2,5F14.5)                                          REA01670
      IER=1002                                                          REA01680
      RETURN                                                            REA01690
      END                                                               REA01700
C     ********************************************************************* REA00010
```

```
C    *                                                                     * REA00020
C    *      SUBROUTINE  READD  IS EXACTLY AS THE SUBROUTINE READF          * REA00030
C    *   WITH THE ONLY DIFFERENSE THAT IT PACKS THE SOLUTION FILE          * REA00040
C    *   INTO THE SUPERSPARSE DATA STRUCTURE.                              * REA00050
C    *                                                                     * REA00060
C    ********************************************************************* REA00070
C                                                                          REA00080
      SUBROUTINE READD(NROWS,NCOL,IER,IFLR,IFLW)                           REA00090
      REAL DATA(7),SOLIST(1)                                               REA00100
      INTEGER*2 CSTAT(15),SOFLAG(1),STAT                                   REA00110
      LOGICAL*1 BITS(16)                                                   REA00120
      COMMON /INIT/ZERO,TINF,IACT,MEMOR                                    REA00130
      COMMON /SOLPAC/SOLFIL(81920)                                         REA00140
      EQUIVALENCE (SOLFIL(5),SOLIST(1),SOFLAG(1))                          REA00150
      DATA CSTAT/2HIN,2HBS,2HLL,2HUL,2HEQ,2HVC,2HSC,2HFC,2HBC,             REA00160
     -2HVR,2HSR,2HRR,2HFR,2HER,2HTR/                                       REA00170
      IER=0                                                                REA00180
      CALL TAPSET (2,IFLR,80,17,1,80)                                      REA00190
      READ(IFLR,100) NROWS,NCOL,IFLGP                                      REA00200
100   FORMAT(I5,30X,I5,10X,I1)                                             REA00210
      IEND=MEMOR-4                                                         REA00220
      II=IEND                                                              REA00230
      NRCOL=NROWS+NCOL                                                     REA00240
      IOFFS=6*NRCOL                                                        REA00250
      INDEX=IOFFS+1                                                        REA00260
      DO 1 I=1,NRCOL                                                       REA00270
      J=I*3-2                                                              REA00280
      DATA(6)=0.                                                           REA00290
      DATA(7)=0.                                                           REA00300
      READ(IFLR,200) SOLIST(J),SOLIST(J+1),STAT,(DATA(L),L=1,5)            REA00310
      IF(IFLGP.NE.0) READ(IFLR,201,END=99) DATA(6),DATA(7)                 REA00320
201   FORMAT(E14.5,16X,E14.5)                                              REA00330
      JW=J                                                                 REA00340
      DO 7 J=1,7                                                           REA00350
      IF(DATA(J).EQ.0..OR.DATA(J).GE.TINF) GO TO 7                         REA00360
      DO 8 K=II,IEND                                                       REA00370
      IF(DATA(J).NE.SOLIST(K)) GO TO 8                                     REA00380
      DATA(J)=K                                                            REA00390
      GO TO 7                                                              REA00400
8     CONTINUE                                                             REA00410
      SOLIST(II)=DATA(J)                                                   REA00420
      DATA(J)=II                                                           REA00430
      II=II-1                                                              REA00440
7     CONTINUE                                                             REA00450
200   FORMAT(2A4,A2,5E14.5)                                                REA00460
      K=I*6                                                                REA00470
      SOFLAG(K)=INDEX-IOFFS-1                                              REA00480
      DO 2 J=1,15                                                          REA00490
```

```
      IF(STAT.NE.CSTAT(J)) GO TO 2                                      REA00500
      ICOD=J-1                                                          REA00510
      GO TO 90                                                          REA00520
2     CONTINUE                                                          REA00530
      IER=1001                                                          REA00540
      RETURN                                                            REA00550
90    IHELP=ICOD                                                        REA00560
      DO 3 J=1,4                                                        REA00570
      BITS(5-J)=.FALSE.                                                 REA00580
      IF(MOD(IHELP,2).EQ.1) BITS(5-J)=.TRUE.                            REA00590
      IHELP=IHELP/2                                                     REA00600
3     CONTINUE                                                          REA00610
      BITS(5)=.TRUE.                                                    REA00620
      IF(DATA(1).EQ.0.) GO TO 10                                        REA00630
      BITS(5)=.FALSE.                                                   REA00640
      SOFLAG(INDEX)=IEND-DATA(1)                                        REA00650
      INDEX=INDEX+1                                                     REA00660
10    BITS(6)=.TRUE.                                                    REA00670
      IF(DATA(2).EQ.0.) GO TO 20                                        REA00680
      BITS(6)=.FALSE.                                                   REA00690
      SOFLAG(INDEX)=IEND-DATA(2)                                        REA00700
      INDEX=INDEX+1                                                     REA00710
20    BITS(7)=.TRUE.                                                    REA00720
      BITS(10)=.FALSE.                                                  REA00730
      IF(DATA(3).EQ.0.) GO TO 30                                        REA00740
      IF(ICOD.EQ.2.OR.ICOD.EQ.4) GO TO 30                               REA00750
      IF(DATA(3).EQ.-TINF) GO TO 25                                     REA00760
      BITS(7)=.FALSE.                                                   REA00770
      SOFLAG(INDEX)=IEND-DATA(3)                                        REA00780
      INDEX=INDEX+1                                                     REA00790
      GO TO 30                                                          REA00800
25    BITS(10)=.TRUE.                                                   REA00810
30    BITS(8)=.TRUE.                                                    REA00820
      BITS(11)=.FALSE.                                                  REA00830
      IF(DATA(4).EQ.0.) GO TO 40                                        REA00840
      IF(ICOD.EQ.3.OR.ICOD.EQ.4) GO TO 40                               REA00850
      IF(DATA(4).EQ.TINF) GO TO 35                                      REA00860
      BITS(8)=.FALSE.                                                   REA00870
      SOFLAG(INDEX)=IEND-DATA(4)                                        REA00880
      INDEX=INDEX+1                                                     REA00890
      GO TO 40                                                          REA00900
35    BITS(11)=.TRUE.                                                   REA00910
40    BITS(9)=.TRUE.                                                    REA00920
      IF(DATA(5).EQ.0.) GO TO 50                                        REA00930
      BITS(9)=.FALSE.                                                   REA00940
      SOFLAG(INDEX)=IEND-DATA(5)                                        REA00950
      INDEX=INDEX+1                                                     REA00960
50    IHELP=10                                                          REA00970
```

```
      IF(DATA(6).EQ.0.) GO TO 60                                        REA00980
      IHELP=30                                                          REA00990
      IF(DATA(6).EQ.TINF) GO TO 60                                      REA01000
      IHELP=20                                                          REA01010
      SOFLAG(INDEX)=IEND-DATA(6)                                        REA01020
      INDEX=INDEX+1                                                     REA01030
60    IHELP1=1                                                          REA01040
      IF(DATA(7).EQ.0.) GO TO 70                                        REA01050
      IHELP1=3                                                          REA01060
      IF(DATA(7).EQ.TINF) GO TO 70                                      REA01070
      IHELP1=2                                                          REA01080
      SOFLAG(INDEX)=IEND-DATA(7)                                        REA01090
      INDEX=INDEX+1                                                     REA01100
70    IHELP=IHELP+IHELP1                                                REA01110
      ICOD=IHELP-11                                                     REA01120
      IF(IHELP.LE.13) GO TO 80                                          REA01130
      ICOD=IHELP-18                                                     REA01140
      IF(IHELP.LE.23) GO TO 80                                          REA01150
      ICOD=IHELP-25                                                     REA01160
80    DO 4 J=1,4                                                        REA01170
      BITS(16-J)=.FALSE.                                                REA01180
      IF(MOD(ICOD,2).EQ.1) BITS(16-J)=.TRUE.                            REA01190
      ICOD=ICOD/2                                                       REA01200
4     CONTINUE                                                          REA01210
      BITS(16)=.FALSE.                                                  REA01220
      IHELP=0                                                           REA01230
      DO 5 J=1,16                                                       REA01240
      IF(BITS(17-J)) IHELP=IHELP+2**(J-1)                               REA01250
5     CONTINUE                                                          REA01260
      SOFLAG(K-1)=IHELP                                                 REA01270
1     CONTINUE                                                          REA01280
      IF(MOD(INDEX,2).NE.0) INDEX=INDEX-1                               REA01290
      INDEX=INDEX/2                                                     REA01300
      DO 11 J=II,IEND                                                   REA01310
      INDEX=INDEX+1                                                     REA01320
      SOLIST(INDEX)=SOLIST(J)                                           REA01330
11    CONTINUE                                                          REA01340
      INDEX=INDEX+4                                                     REA01350
      WRITE(6,300) INDEX                                                REA01360
300   FORMAT(5X,'WORDS REQUIRED FOR PACKED FILE: ',I6)                  REA01370
      SOLFIL(1)=INDEX                                                   REA01380
      SOLFIL(2)=0                                                       REA01390
      SOLFIL(3)=NROWS                                                   REA01400
      SOLFIL(4)=NCOL                                                    REA01410
      WRITE(IFLW'1) (SOLFIL(I),I=1,INDEX)                               REA01420
      RETURN                                                            REA01430
99    WRITE(6,999) SOLIST(JW),SOLIST(JW+1),STAT,(DATA(L),L=1,5)         REA01440
999   FORMAT(2X,2A4,A2,5F14.5)                                          REA01450
```

```
      IER=1002                                                          REA01460
      RETURN                                                            REA01470
      END                                                               REA01480
C     ****************************************************************  INT00010
C     *                                                              *  INT00020
C     *      SUBROUTINE  INTERP  EXEQUTES THE QUERY USING THE CODE   *  INT00030
C     *   GENERATED BY THE PARSE SUBROUTINE.                         *  INT00040
C     *      CODE - THE CODE GENERATED BY THE PARSER                 *  INT00050
C     *      MASK - THE MASK FIELD OF THE QUERY                      *  INT00060
C     *      ARG2, ARG4 - THE VALUES OF ARG1, ARG2 OF CONDITION FIELD*  INT00070
C     *      K1, K2, K3 - AS THEY ARE SET BY THE SET COMMAND         *  INT00080
C     *      K1=START    K2=END     K3=STEP                          *  INT00090
C     *                                                              *  INT00100
C     ****************************************************************  INT00110
C                                                                       INT00120
      SUBROUTINE INTERP(CODE,MASK,ARG2,ARG4,K1,K2,K3,NROWS,NCOL,BLANK,ST INT00130
     -AR,IER,IFLG)                                                      INT00140
      COMMON /INIT/ZERO,TINF,IACT                                       INT00150
      INTEGER*2 BLANK,STAR,CODE(20),NAME(8),MASK(8),SNAME(8),SOFLAG(1)  INT00160
      COMMON /SOLPAC/SOLFIL(81920)                                      INT00170
      EQUIVALENCE (SOLFIL(5),SOLIST(1),SOFLAG(1))                       INT00180
      INTEGER PROP(7)                                                   INT00190
      REAL SOLIST(1),RECORD(7),RECAD(7)                                 INT00200
      LOGICAL EVCOD,EVMN                                                INT00210
      IER=0                                                             INT00220
C                                                                       INT00230
C     SET UP INFORMATIONS FOR PRINT RESULTS.                            INT00240
C     INITIALIZE VALUES FOR MINIMUM AND MAXIMUM.                        INT00250
      IPRINT=2                                                          INT00260
      SMIN=.7273E74                                                     INT00270
      SMAX=-SMIN                                                        INT00280
      DO 4 J=1,7                                                        INT00290
      RECAD(J)=0.                                                       INT00300
      PROP(J)=0.                                                        INT00310
      IF(CODE(11).EQ.0) PROP(J)=J                                       INT00320
4     CONTINUE                                                          INT00330
      IF(CODE(11).EQ.0) GO TO 12                                        INT00340
      DO 5 I=1,7                                                        INT00350
      JSAVE=0                                                           INT00360
      DO 6 J=12,18                                                      INT00370
      IF(CODE(J).NE.I) GO TO 6                                          INT00380
      JSAVE=J                                                           INT00390
      GO TO 11                                                          INT00400
6     CONTINUE                                                          INT00410
11    IF(JSAVE.EQ.0) GO TO 12                                           INT00420
      PROP(I)=JSAVE-11                                                  INT00430
5     CONTINUE                                                          INT00440
C                                                                       INT00450
```

```
C    SET UP LIMITS FOR SEARCHING.                                                    INT00460
C                                                                                    INT00470
12     NEND=K2                                                                       INT00480
       NCUR=K1                                                                       INT00490
       IF(CODE(2).EQ.3.AND.K2.GT.NROWS) NEND=NROWS                                   INT00500
       IF(CODE(2).NE.2) GO TO 91                                                     INT00510
       NCUR=NROWS+K3+K1-1                                                            INT00520
       NEND=NROWS+K2                                                                 INT00530
       IF(NEND.GT.(NROWS+NCOL)) NEND=NROWS+NCOL                                      INT00540
91     IF(CODE(2).EQ.3) IPRINT=3                                                     INT00550
       NCUR=NCUR-K3                                                                  INT00560
       ICOUNT=0                                                                      INT00570
99     NCUR=NCUR+K3                                                                  INT00580
       IF(NCUR.LE.NEND) GO TO 10                                                     INT00590
C                                                                                    INT00600
C    IF END OF SEARCHING PRINT BOTTOM LINE.                                          INT00610
C                                                                                    INT00620
       IF(CODE(1).NE.17.OR.CODE(2).NE.1) GO TO 105                                   INT00630
       IACT=0                                                                        INT00640
       WRITE(6,500)                                                                  INT00650
500    FORMAT(10X,' F I L E   D E A C T I V E ',//)                                  INT00660
       RETURN                                                                        INT00670
105    IF(CODE(4).NE.9.AND.CODE(5).GT.6)CALL PRRES(CODE,SNAME,RECAD,ICODMINT00680
      -X,ICOUNT,PROP)                                                                INT00690
90     CALL PRBOT(CODE,RECORD,RECAD,ICOUNT,MASK,PRCP,IPRINT)                         INT00700
       RETURN                                                                        INT00710
10     IF(IACT.EQ.0) GO TO 102                                                       INT00720
C                                                                                    INT00730
C    CHECK IF ACTIVE FILE EXISTS.                                                    INT00740
C    IF IT IS CHECK IF CURRENT RECORD ACTIVE.                                        INT00750
C    IF NOT GET THE NEXT RECORD IF ANY.                                              INT00760
C                                                                                    INT00770
       IHELP=SOFLAG(NCUR*6-1)                                                        INT00780
       IF(MOD(IHELP,2).EQ.0.AND.CODE(1).NE.16) GO TO 99                              INT00790
       IF(MOD(IHELP,2).EQ.1.AND.CODE(1).EQ.16) GO TO 99                              INT00800
       IF(CODE(1).NE.17.OR.CODE(2).NE.1) GO TO 102                                   INT00810
       SOFLAG(NCUR*6-1)=SOFLAG(NCUR*6-1)-1                                           INT00820
       GO TO 99                                                                      INT00830
C                                                                                    INT00840
C    PUT BLANKS TO THE NAME.                                                         INT00850
C                                                                                    INT00860
102    DO 2 J=1,8                                                                    INT00870
       NAME(J)=BLANK                                                                 INT00880
2      CONTINUE                                                                      INT00890
C                                                                                    INT00900
C    GET THE NAME OF CURRENT RECORD.                                                 INT00910
C                                                                                    INT00920
       CALL CURNAM(MASK,NAME,NCUR,BLANK,STAR)                                        INT00930
```

```
C                                                                       INT00940
C    CHECK IF NAME MATCHES TO THE MASK.                                 INT00950
C    IF NOT GET THE NEXT RECORD.                                        INT00960
C                                                                       INT00970
70     DO 1 I=1,8                                                       INT00980
       IF(MASK(I).EQ.STAR) GO TO 1                                      INT00990
       IF(MASK(I).NE.NAME(I)) GO TO 99                                  INT01000
1      CONTINUE                                                         INT01010
C                                                                       INT01020
C    IF IFLG=0 (SPARSE STRUCTURE) CALL DEPACK TO GET                    INT01030
C    THE VALUES OF CURRENT RECORD ELSE CALL DPAC.                       INT01040
C                                                                       INT01050
20     IF(IFLG.EQ.0) GO TO 201                                          INT01060
       CALL DEPACK(CODE,ICOD,RECORD,NCUR,BLANK,NROWS,NCOL,IER)          INT01070
       GO TO 202                                                        INT01080
201    CALL DPAC(CODE,ICOD,RECORD,NCUR,BLANK,NROWS,NCOL,IER)            INT01090
202    IF(IER.NE.0) RETURN                                              INT01100
C                                                                       INT01110
C    IF NO CONDITION FIELD SKIP                                         INT01120
C    ELSE EVALUATE CONDITION USING EVCOD AND EVMN.                      INT01130
C                                                                       INT01140
       IF(CODE(4).EQ.0) GO TO 60                                        INT01150
       IF(CODE(4).EQ.9) GO TO 50                                        INT01160
       IF(CODE(5).LT.7) GO TO 21                                        INT01170
       IF(CODE(5).NE.7) GO TO 22                                        INT01180
C                                                                       INT01190
C    EVALUATE BOUND CONDITION. (MAX OR MIN)                             INT01200
C                                                                       INT01210
       IF(RECORD(CODE(4)).GT.SMIN) GO TO 99                             INT01220
       IF(RECORD(CODE(4)).EQ.SMIN) GO TO 24                             INT01230
       SMIN=RECORD(CODE(4))                                             INT01240
       GO TO 23                                                         INT01250
22     IF(RECORD(CODE(4)).LT.SMAX) GO TO 99                             INT01260
       IF(RECORD(CODE(4)).EQ.SMAX) GO TO 24                             INT01270
       SMAX=RECORD(CODE(4))                                             INT01280
23     ICOUNT=0                                                         INT01290
       DO 7 I=1,7                                                       INT01300
       RECAD(I)=RECORD(I)                                               INT01310
       SNAME(I)=NAME(I)                                                 INT01320
7      CONTINUE                                                         INT01330
       SNAME(8)=NAME(8)                                                 INT01340
       ICODMX=ICOD                                                      INT01350
24     ICOUNT=ICOUNT+1                                                  INT01360
       GO TO 99                                                         INT01370
21     IF(EVCOD(CODE,ICOD,RECORD,ARG2,ARG4,IER)) GO TO 60               INT01380
       GO TO 99                                                         INT01390
50     INDEX=5                                                          INT01400
       IF(.NOT.EVMN(CODE,ICOD,RECORD,INDEX,MIN,MAX,ICOUNT,IER)) GO TO 99 INT01410
```

```
C                                                                       INT01420
C    ADD 1 TO THE COUNTER OF QUALIFIED RECORDS.                         INT01430
C                                                                       INT01440
60     ICOUNT=ICOUNT+1                                                  INT01450
       IF(CODE(1).LE.15) GO TO 301                                      INT01460
C                                                                       INT01470
C    PROCESS HERE THA ACTIVE OR DEACTIVE COMMANDS.                      INT01480
C                                                                       INT01490
       IHELP=SOFLAG(NCUR*6-1)                                           INT01500
       IHELP=MOD(IHELP,2)                                               INT01510
       IF(IHELP.EQ.1.AND.CODE(1).EQ.16) GO TO 99                        INT01520
       IF(IHELP.EQ.0.AND.CODE(1).EQ.17) GO TO 99                        INT01530
       IF(IHELP.EQ.1.AND.CODE(1).EQ.17) SOFLAG(NCUR*6-1)=SOFLAG(NCUR*6-1)INT01540
      --1                                                               INT01550
       IF(IHELP.EQ.0.AND.CODE(1).EQ.16) SOFLAG(NCUR*6-1)=SOFLAG(NCUR*6-1)INT01560
      -+1                                                               INT01570
       GO TO 99                                                         INT01580
301    IF(CODE(1).NE.12) GO TO 30                                       INT01590
       IF(CODE(3).NE.3) GO TO 99                                        INT01600
       GO TO 90                                                         INT01610
30     IF(CODE(1).EQ.11.OR.CODE(1).EQ.15) GO TO 40                      INT01620
C                                                                       INT01630
C    ADDVALUES IF ADD OR AVERAGE COMMANDS.                              INT01640
C                                                                       INT01650
       DO 3 J=1,3                                                       INT01660
       L=J                                                              INT01670
       IF(L.EQ.3) L=5                                                   INT01680
       RECAD(L)=RECAD(L)+RECORD(L)                                      INT01690
3      CONTINUE                                                         INT01700
       IF(CODE(3).NE.3) GO TO 99                                        INT01710
       GO TO 90                                                         INT01720
40     IF(ICOUNT.EQ.1) CALL PRHEAD(CODE,PROP,IPRINT)                    INT01730
C                                                                       INT01740
C    PRINT HEADINGS IF FIRST RECORD.                                    INT01750
C    PRINT THE RECORD AS SPECIFIED BY PRINT FIELD.                      INT01760
C                                                                       INT01770
       CALL PRRES(CODE,NAME,RECORD,ICOD,NCUR,PROP)                      INT01780
C      WRITE(6,100) (RECORD(J),J=1,7),ICOUNT                            INT01790
COO    FORMAT(2X,7F10.3,I3)                                             INT01800
       IF(IER.NE.0) RETURN                                              INT01810
       IF(CODE(3).NE.3)GO TO 99                                         INT01820
C                                                                       INT01830
C    IF NO * AT THE MASK NAME UNIQUE EXIT.                              INT01840
C                                                                       INT01850
       GO TO 90                                                         INT01860
       END                                                              INT01870
C    ********************************************************************CUR00010
C    *                                                                  * CUR00020
```

```
C    *      SUBROUTINE  CURNAM  RETURNS THE NAME OF THE CURRENT          * CUR00030
C    *   RECORD AT THE ARRAY 'NAME'.                                     * CUR00040
C    ********************************************************************* CUR00050
C                                                                         CUR00060
      SUBROUTINE CURNAM(MASK,NAME,NCUR,BLANK,STAR)                        CUR00070
      COMMON /SOLPAC/SOLFIL(81920)                                        CUR00080
      REAL PACK(1)                                                        CUR00090
      LOGICAL*1 NAME(16),LOGPAC(1)                                        CUR00100
      INTEGER*2 MASK(8),BLANK,STAR                                        CUR00110
      EQUIVALENCE (SOLFIL(5),PACK(1),LOGPAC(1))                           CUR00120
      INDEX=NCUR*12-12                                                    CUR00130
      DO 1 I=2,16,2                                                       CUR00140
      J=I/2                                                               CUR00150
      NAME(I-1)=LOGPAC(INDEX+J)                                           CUR00160
1     CONTINUE                                                            CUR00170
      RETURN                                                              CUR00180
      END                                                                 CUR00190
C    ********************************************************************* DEP00010
C    *                                                                   * DEP00020
C    *      SUBROUTINE  DEPACK RETURNS AT ARRAY RECORD EXPLICITLY        * DEP00030
C    *   ALL THE VALUES OF THE CURRENT RECORD AND THE STATUS AT ICOD     * DEP00040
C    *   IT IS USED TO GET THE VALUES FROM THE SPARSE DATA STRUCTURE.    * DEP00050
C    *                                                                   * DEP00060
C    ********************************************************************* DEP00070
C                                                                         DEP00080
      SUBROUTINE DEPACK(CODE,ICOD,RECORD,NCUR,BLANK,NROWS,NCOL,IER)       DEP00090
      REAL RECORD(7),SOLIST(1)                                            DEP00100
      INTEGER*2 CODE(20),SOFLAG(1)                                        DEP00110
      LOGICAL*1 BITS(16),SONAME(1)                                        DEP00120
      COMMON /INIT/ZERO,TINF,IACT                                         DEP00130
      COMMON /SOLPAC/SOLFIL(81920)                                        DEP00140
      EQUIVALENCE (SOLFIL(5),SOLIST(1),SOFLAG(1),SONAME(1))               DEP00150
      IER=0                                                               DEP00160
C                                                                         DEP00170
C    CREATE THE BIT PATTERN.                                              DEP00180
C                                                                         DEP00190
      IOFFS=3*(NROWS+NCOL)+1                                              DEP00200
      IBITS=NCUR*6-1                                                      DEP00210
      IHELP=SOFLAG(IBITS)                                                 DEP00220
C                                                                         DEP00230
C    IF BITS NEGATIVE MAKE IT POSITIVE                                    DEP00240
C                                                                         DEP00250
      IF(IHELP.LT.0) IHELP=IHELP+65536                                    DEP00260
      DO 1 I=1,16                                                         DEP00270
      BITS(17-I)=.FALSE.                                                  DEP00280
      IF(MOD(IHELP,2).EQ.1) BITS(17-I)=.TRUE.                             DEP00290
      IHELP=IHELP/2                                                       DEP00300
1     CONTINUE                                                            DEP00310
```

```
      IHELP=0                                                                DEP00320
      ICODP=0                                                                DEP00330
      DO 2 I=1,4                                                             DEP00340
      IF(BITS(5-I)) IHELP=IHELP+2**(I-1)                                     DEP00350
      IF(BITS(16-I)) ICODP=ICODP+2**(I-1)"                                   DEP00360
2     CONTINUE                                                               DEP00370
      ICOD=IHELP                                                             DEP00380
      INDEX=IOFFS+SOFLAG(IBITS+1)                                            DEP00390
      IF(SOFLAG(IBITS+1).LT.0) INDEX=INDEX+65536                             DEP00400
      DO 3 I=1,5                                                             DEP00410
      RECORD(I)=0.0                                                          DEP00420
      IF(I.EQ.3.AND.(ICOD.EQ.2.OR.ICOD.EQ.4)) GO TO 10                       DEP00430
      IF(I.EQ.4.AND.(ICOD.EQ.3.OR.ICOD.EQ.4)) GO TO 10                       DEP00440
      IF(BITS(4+I)) GO TO 3                                                  DEP00450
      RECORD(I)=SOLIST(INDEX)                                                DEP00460
      INDEX=INDEX+1                                                          DEP00470
      GO TO 3                                                                DEP00480
10    RECORD(I)=RECORD(1)                                                    DEP00490
3     CONTINUE                                                               DEP00500
      IF(BITS(10)) RECORD(3)=-TINF                                           DEP00510
      IF(BITS(11)) RECORD(4)=TINF                                            DEP00520
      RECORD(6)=0.                                                           DEP00530
      IF(ICODP.LE.2) GO TO 20                                                DEP00540
      IF(ICODP.GT.5) GO TO 30                                                DEP00550
      RECORD(6)=SOLIST(INDEX)                                                DEP00560
      INDEX=INDEX+1                                                          DEP00570
      GO TO 20                                                               DEP00580
30    RECORD(6)=TINF                                                         DEP00590
20    RECORD(7)=0.                                                           DEP00600
      IF(ICODP.EQ.0.OR.ICODP.EQ.3.OR.ICODP.EQ.6) RETURN                      DEP00610
      IF(ICODP.EQ.2.OR.ICODP.EQ.5.OR.ICODP.EQ.8) GO TO 40                    DEP00620
      RECORD(7)=SOLIST(INDEX)                                                DEP00630
      RETURN                                                                 DEP00640
40    RECORD(7)=TINF                                                         DEP00650
      RETURN                                                                 DEP00660
      END                                                                    DEP00670
C     ************************************************************************ DPA00010
C     *                                                                      * DPA00020
C     *      SUBROUTINE  DPAC  RETURNS THE SAME VALUES AS DEPACK             * DPA00030
C     *   BUT USING THE SUPERSPARSE DATA STRUCTURE.                          * DPA00040
C     ************************************************************************ DPA00050
C                                                                            DPA00060
      SUBROUTINE DPAC(CODE, ICOD,RECORD,NCUR,BLANK,NROWS,NCOL,IER)           DPA00070
      REAL RECORD(7),SOLIST(1)                                               DPA00080
      INTEGER*2 CODE(20),SOFLAG(1)                                           DPA00090
      LOGICAL*1 BITS(16),SONAME(1)                                           DPA00100
      COMMON /INIT/ZERO,TINF,IACT,MEMOR,ISIZE                                DPA00110
      COMMON /SOLPAC/SOLFIL(81920)                                           DPA00120
```

```
      EQUIVALENCE (SOLFIL(5),SOLIST(1),SOFLAG(1),SONAME(1))                DPA00130
      IER=0                                                                DPA00140
      IOFFS=6*(NROWS+NCOL)+1                                               DPA00150
      IEND=ISIZE-4                                                         DPA00160
      IBITS=NCUR*6-1                                                       DPA00170
      IHELP=SOFLAG(IBITS)                                                  DPA00180
      IF(IHELP.LT.0) IHELP=IHELP+65536                                     DPA00190
      DO 1 I=1,16                                                          DPA00200
      BITS(17-I)=.FALSE.                                                   DPA00210
      IF(MOD(IHELP,2).EQ.1) BITS(17-I)=.TRUE.                              DPA00220
      IHELP=IHELP/2                                                        DPA00230
1     CONTINUE                                                             DPA00240
      IHELP=0                                                              DPA00250
      ICODP=0                                                              DPA00260
      DO 2 I=1,4                                                           DPA00270
      IF(BITS(5-I)) IHELP=IHELP+2**(I-1)                                   DPA00280
      IF(BITS(16-I)) ICODP=ICODP+2**(I-1)"                                 DPA00290
2     CONTINUE                                                             DPA00300
      ICOD=IHELP                                                           DPA00310
      INDEX=IOFFS+SOFLAG(IBITS+1)                                          DPA00320
      IF(SOFLAG(IBITS+1).LT.0) INDEX=INDEX+65536                           DPA00330
      DO 3 I=1,5                                                           DPA00340
      RECORD(I)=0.0                                                        DPA00350
      IF(I.EQ.3.AND.(ICOD.EQ.2.OR.ICOD.EQ.4)) GO TO 10                     DPA00360
      IF(I.EQ.4.AND.(ICOD.EQ.3.OR.ICOD.EQ.4)) GO TO 10                     DPA00370
      IF(BITS(4+I)) GO TO 3                                                DPA00380
      RECORD(I)=SOLIST(IEND-SOFLAG(INDEX))                                 DPA00390
      INDEX=INDEX+1                                                        DPA00400
      GO TO 3                                                              DPA00410
10    RECORD(I)=RECORD(1)                                                  DPA00420
3     CONTINUE                                                             DPA00430
      IF(BITS(10)) RECORD(3)=-TINF                                         DPA00440
      IF(BITS(11)) RECORD(4)=TINF                                          DPA00450
      RECORD(6)=0.                                                         DPA00460
      IF(ICODP.LE.2) GO TO 20                                              DPA00470
      IF(ICODP.GT.5) GO TO 30                                              DPA00480
      RECORD(6)=SOLIST(IEND-SOFLAG(INDEX))                                 DPA00490
      INDEX=INDEX+1                                                        DPA00500
      GO TO 20                                                             DPA00510
30    RECORD(6)=TINF                                                       DPA00520
20    RECORD(7)=0.                                                         DPA00530
      IF(ICODP.EQ.0.OR.ICODP.EQ.3.OR.ICODP.EQ.6) RETURN                    DPA00540
      IF(ICODP.EQ.2.OR.ICODP.EQ.5.OR.ICODP.EQ.8) GO TO 40                  DPA00550
      RECORD(7)=SOLIST(IEND-SOFLAG(INDEX))                                 DPA00560
      RETURN                                                               DPA00570
40    RECORD(7)=TINF                                                       DPA00580
      RETURN                                                               DPA00590
      END                                                                  DPA00600
```

```
C    ************************************************************************ EVC00010
C    *                                                                      * EVC00020
C    *     LOGICAL FUNCTION  EVCOD  RETURNS 'TRUE' IF CONDITION FIELD       * EVC00030
C    *   IS SATISFIED BY THE CURRENT RECORD AND FALSE OTHERWISE.            * EVC00040
C    ************************************************************************ EVC00050
C                                                                             EVC00060
      LOGICAL FUNCTION EVCOD(CODE,ICOD,RECORD,ARG2,ARG4,IER)                  EVC00070
      COMMON /INIT/ZERO,TINF,IACT                                             EVC00080
      INTEGER*2 CODE(20)                                                      EVC00090
      REAL RECORD(7)                                                          EVC00100
      LOGICAL EVMN                                                            EVC00110
      IER=0                                                                   EVC00120
      EVCOD=.FALSE.                                                           EVC00130
      INDEX=5                                                                 EVC00140
      ARG=ARG2                                                                EVC00150
30    IHELP=CODE(INDEX)                                                       EVC00160
      IF(CODE(INDEX+1).NE.9) ARG=RECORD(CODE(INDEX+1))                        EVC00170
      GO TO (1,2,3,4,5,6,7,7), IHELP                                          EVC00180
1     IF(ABS(RECORD(CODE(INDEX-1))-ARG).LE.ZERO) EVCOD=.TRUE.                 EVC00190
      GO TO 10                                                                EVC00200
2     IF(RECORD(CODE(INDEX-1)).NE.ARG) EVCOD=.TRUE.                           EVC00210
      GO TO 10                                                                EVC00220
3     IF(RECORD(CODE(INDEX-1)).GT.ARG) EVCOD=.TRUE.                           EVC00230
      GO TO 10                                                                EVC00240
4     IF(RECORD(CODE(INDEX-1)).GE.ARG) EVCOD=.TRUE.                           EVC00250
      GO TO 10                                                                EVC00260
5     IF(RECORD(CODE(INDEX-1)).LT.ARG) EVCOD=.TRUE.                           EVC00270
      GO TO 10                                                                EVC00280
6     IF(RECORD(CODE(INDEX-1)).LE.ARG) EVCOD=.TRUE.                           EVC00290
7     CONTINUE                                                                EVC00300
10    IF(CODE(7).EQ.0) RETURN                                                 EVC00310
      IF(INDEX.NE.5) RETURN                                                   EVC00320
      IF(CODE(7).EQ.1.AND.EVCOD) RETURN                                       EVC00330
      IF(CODE(7).EQ.2.AND..NOT.EVCOD) RETURN                                  EVC00340
20    EVCOD=.FALSE.                                                           EVC00350
      INDEX=9                                                                 EVC00360
      ARG=ARG4                                                                EVC00370
      IF(CODE(8).NE.9) GO TO 30                                               EVC00380
      EVCOD=EVMN(CODE,ICOD,RECORD,INDEX,MIN,MAX,ICOUNT,IER)                   EVC00390
      RETURN                                                                  EVC00400
      END                                                                     EVC00410
C    ************************************************************************ EVM00010
C    *                                                                      * EVM00020
C    *     LOGICAL FUNCTION  EVMN  RETURNS 'TRUE' IF THE STATUS OF THE      * EVM00030
C    *  CURRENT RECORD IS THE SAME AS IN CONDITION FIELD OF QUERY           * EVM00040
C    *  AND 'FALSE' OTHERWISE.                                              * EVM00050
C    ************************************************************************ EVM00060
C                                                                             EVM00070
```

```
      LOGICAL FUNCTION EVMN(CODE,ICOD,RECORD,INDEX,MIN,MAX,ICOUNT,IER)  EVM00080
      INTEGER*2 CODE(20)                                                EVM00090
      REAL RECORD(7)                                                    EVM00100
      IER=0                                                             EVM00110
      EVMN=.FALSE.                                                      EVM00120
      IF(CODE(INDEX).EQ.ICOD) EVMN=.TRUE.                               EVM00130
      RETURN                                                            EVM00140
      END                                                               EVM00150
C     ****************************************************************  PRH00010
C     *                                                              *  PRH00020
C     *     SUBROUTINE  PRHEAD  PRINT HEADINGS FOR COLUMNS IF IPRINT=3  *  PRH00030
C     *  OR HEADINGS FOR ROWS IF IPRINT=2.                           *  PRH00040
C     ****************************************************************  PRH00050
C                                                                         PRH00060
      SUBROUTINE PRHEAD(CODE,PROP,IPRINT)                               PRH00070
      INTEGER*2 CODE(20)                                                PRH00080
      INTEGER HEAD(36),BUFF(33),PROP(7)                                 PRH00090
      DATA BUFF/4HNUMB,4HER  ,4H.NAM,4HE.. ,4H AT ,28*'    '/           PRH00100
      DATA HEAD/4H ...,4HACTI,4HVITY,4H... ,4H ..I,4HNPUT,4H COS,4HT.. ,PRH00110
     -4H ..L,4HOWER,4H LIM,4HIT. ,4H ..U,4HPPER,4H LIM,4HIT. ,4H .RE,4HDPRH00120
     -UCE,4HD CO,4HST. ,4H .UP,4HPER ,4HPENA,4HLTY ,4H .DO,4HWN P,4HENALPRH00130
     -,4HTY. ,4H SLA,4HCK A,4HCTIV,4HITY ,4H .DU,4HAL A,4HCTIV,4HITY /  PRH00140
      WRITE(6,100)                                                      PRH00150
100   FORMAT(2X,'THE FOLLOWING ROWS OR COLUMNS SATISFY CONDITIONS',/)   PRH00160
      DO 1 I=1,7                                                        PRH00170
      ISAVE=I                                                           PRH00180
      IF(PROP(I).EQ.0) GO TO 10                                         PRH00190
      INDEX=PROP(I)*4-4                                                 PRH00200
      IF(IPRINT.NE.3) GO TO 11                                          PRH00210
      IF(PROP(I).EQ.2) INDEX=28                                         PRH00220
      IF(PROP(I).EQ.5) INDEX=32                                         PRH00230
11    IHELP=I*4+1                                                       PRH00240
      DO 1 J=1,4                                                        PRH00250
      BUFF(IHELP+J)=HEAD(INDEX+J)                                       PRH00260
1     CONTINUE                                                          PRH00270
      ISAVE=ISAVE+1                                                     PRH00280
10    ISAVE=(ISAVE-1)*4+5                                               PRH00290
      WRITE(6,200) (BUFF(I),I=1,ISAVE)                                  PRH00300
200   FORMAT(1H1,33A4)                                                  PRH00310
      RETURN                                                            PRH00320
      END                                                               PRH00330
C     ****************************************************************  PRR00010
C     *                                                              *  PRR00020
C     *     SUBROUTINE  PRRES  PRINTS THE QUALIFIED RECORDS ACCORDING   *  PRR00030
C     *  TO THE PRINT FIELD OF THE QUERY.                            *  PRR00040
C     *  IT USES VARIABLE OBJECT CODE FORMAT.                        *  PRR00050
C     ****************************************************************  PRR00060
C                                                                         PRR00070
```

```
      SUBROUTINE PRRES(CODE,NAME,RECORD,ICOD,ICOUNT,PROP)                PRR00080
      INTEGER*2 NAME(8),CODE(20),HDCOD(15)                               PRR00090
      INTEGER PROP(7),FMT1(18),FMT2(18),FMT3(18),FMTN(7)                 PRR00100
      REAL RECORD(7)                                                     PRR00110
      TINF=0.1E74                                                        PRR00120
      DATA HDCOD/2HIN,2HBS,2HLL,2HUL,2HEQ,2HVC,2HSC,2HFC,2HBC,2HVR,      PRR00130
     -2HSR,2HRR,2HFR,2HER,2HTR/                                          PRR00140
      DATA FMT1/4H(1H+,4H,  T,4H    ,4H,2X,,4HF14.,4H5)  /               PRR00150
      DATA FMT2/4H(1H+,4H,  T,4H    ,4H,2X,,4H'   ,4H    ,4H   N,       PRR00160
     -4HONE',4H)   /                                                     PRR00170
      DATA FMT3/4H(1H+,4H,  T,4H    ,4H,2X,,4H'   ,4H   I,4HNFIN,        PRR00180
     -4HITY',4H)   /                                                     PRR00190
      DATA FMTN/7*4H2   /                                                PRR00200
C     DATA FMTN/4H21  ,4H37  ,4H53  ,4H69  ,4H85  ,4H101 ,4H117 /        PRR00210
      IF(CODE(1).NE.13.AND.CODE(1).NE.14) GO TO 4C                       PRR00220
      WRITE(6,300)                                                       PRR00230
300   FORMAT(1H+,' SUMS OR AVERAGES :  ')                                PRR00240
      GO TO 41                                                           PRR00250
40    WRITE(6,100) ICOUNT,(NAME(I),I=1,8),HDCOD(ICOD+1)                  PRR00260
100   FORMAT(1H+,I5,2X,8A1,2X,A2)                                        PRR00270
41    DO 11 I=1,7                                                        PRR00280
      IHELP=PROP(I)                                                      PRR00290
      IF(IHELP.EQ.0) GO TO 10                                            PRR00300
      GO TO (1,1,2,2,1,3,3),IHELP                                        PRR00310
1     FMT1(3)=FMTN(I)                                                    PRR00320
      IF(RECORD(IHELP).EQ.0.0) RECORD(IHELP)=0.00C0001                   PRR00330
      WRITE(6,FMT1) RECORD(IHELP)                                        PRR00340
      GO TO 11                                                           PRR00350
2     FMT1(3)=FMTN(I)                                                    PRR00360
      IF(ABS(RECORD(IHELP)).GT.TINF) GO TO 21                            PRR00370
      IF(RECORD(IHELP).EQ.0.0) RECORD(IHELP)=0.0000001                   PRR00380
      WRITE(6,FMT1) RECORD(IHELP)                                        PRR00390
400   FORMAT(1H+,A4)                                                     PRR00400
      GO TO 11                                                           PRR00410
21    FMT2(3)=FMTN(I)                                                    PRR00420
      WRITE(6,FMT2)                                                      PRR00430
      GO TO 11                                                           PRR00440
3     FMT1(3)=FMTN(I)                                                    PRR00450
      IF(RECORD(IHELP).EQ.0.) GO TO 31                                   PRR00460
      IF(RECORD(IHELP).GT.TINF) GO TO 32                                 PRR00470
      IF(RECORD(IHELP).EQ.0.0) RECORD(IHELP)=0.0000001                   PRR00480
      WRITE(6,FMT1) RECORD(IHELP)                                        PRR00490
      GO TO 11                                                           PRR00500
31    FMT2(3)=FMTN(I)                                                    PRR00510
      WRITE(6,FMT2)                                                      PRR00520
      GO TO 11                                                           PRR00530
32    FMT3(3)=FMTN(I)                                                    PRR00540
      WRITE(6,FMT3)                                                      PRR00550
```

```
11    CONTINUE                                                          PRR00560
10    WRITE(6,200)                                                      PRR00570
200   FORMAT(1H )                                                       PRR00580
      RETURN                                                            PRR00590
      END                                                               PRR00600
C     ****************************************************************  PRB00010
C     *                                                              *  PRB00020
C     *     SUBROUTINE  PRBOT  PRINTS THE TOTAL NUMBER OF QUALIFIED  *  PRB00030
C     *  RECORDS AFTER THE LAST RECORD PRINTED BY PRRES SUBROUTINE.  *  PRB00040
C     *  ALSO IT PRINTS THE RESULTS FROM ADD AND AVERAGE COMMANDS.   *  PRB00050
C     *                                                              *  PRB00060
C     ****************************************************************  PRB00070
C                                                                       PRB00080
      SUBROUTINE PRBOT(CODE,RECORD,RECAD,ICOUNT,MASK,PROP,IPRINT)       PRB00090
      COMMON /INIT/ZERO,TINF,IACT                                       PRB00100
      INTEGER*2 CODE(20),MASK(8)                                        PRB00110
      INTEGER PROP(7)                                                   PRB00120
      REAL RECORD(7),RECAD(7)                                           PRB00130
      IF(CODE(1).GT.15) GO TO 30                                        PRB00140
C                                                                       PRB00150
C     IF ADD OR AVERAGE COMMANDS SKIP TO 10                             PRB00160
C                                                                       PRB00170
      IF(CODE(1).EQ.13.OR.CODE(1).EQ.14) GO TO 10                       PRB00180
200   FORMAT(2X,7F10.3)                                                 PRB00190
21    WRITE(6,100) ICOUNT,(MASK(I),I=1,8)                               PRB00200
100   FORMAT(2X,/,2X,I5,2X,'ROWS OR COLUMNS WITH MASK : ',8A1,/,2X,'SATIPRB00210
     -SFY THE CONDITIONS')                                              PRB00220
      RETURN                                                            PRB00230
10    IF(CODE(1).EQ.13) GO TO 20                                        PRB00240
      DO 1 J=1,7                                                        PRB00250
      RECAD(J)=RECAD(J)/ICOUNT                                          PRB00260
1     CONTINUE                                                          PRB00270
20    PROP(1)=1                                                         PRB00280
      PROP(2)=2                                                         PRB00290
      PROP(3)=5                                                         PRB00300
      PROP(4)=0                                                         PRB00310
      CALL PRHEAD(CODE,PROP,IPRINT)                                     PRB00320
      CALL PRRES(CODE,MASK,RECAD,ICOD,ICOUNT,PROP)                      PRB00330
      GO TO 21                                                          PRB00340
30    IF(CODE(1).EQ.16) GO TO 31                                        PRB00350
      IACT=IACT-ICOUNT                                                  PRB00360
      WRITE(6,300) ICOUNT,IACT                                          PRB00370
300   FORMAT(5X,//,5X,I6,'   RECORDS  DEACTIVATED',/,5X,' TOTAL RECORDS PRB00380
     - ACTIVE :',I6)                                                    PRB00390
      RETURN                                                            PRB00400
31    IACT=IACT+ICOUNT                                                  PRB00410
      WRITE(6,400) ICOUNT,IACT                                          PRB00420
400   FORMAT(5X,/,5X,I6,'   RECORDS ACTIVATED',/,5X,' TOTAL RECORDS ACTIPRB00430
```

```
     -VE :',I6)                                                          PRB00440
      RETURN                                                             PRB00450
      END                                                                PRB00460
C     ************************************************************     ERR00010
C     *                                                          *     ERR00020
C     *     SUBROUTINE  ERROR  PRINTS MESSAGES FOR SYNTAX OR EXCECUTICN *ERR00030
C     *  ERRORS.                                                 *     ERR00040
C     ************************************************************     ERR00050
C                                                                        ERR00060
      SUBROUTINE ERROR(IER)                                              ERR00070
      IF(IER.GT.1000) GO TO 999                                          ERR00080
      WRITE(6,1000)                                                      ERR00090
1000  FORMAT(5X,'***  SYNTAX   ERROR  ***')                             ERR00100
      INDEX=IER/100                                                      ERR00110
      IHELP=MOD(IER,100)                                                 ERR00120
      IER=0                                                              ERR00130
      IF(INDEX.NE.0) GO TO 10                                            ERR00140
      WRITE(6,1)                                                         ERR00150
1     FORMAT(5X,'NO BLANK QUERY IS ACCAPTED')                            ERR00160
      RETURN                                                             ERR00170
10    GO TO (100,200,300,300,500,600,700),INDEX                          ERR00180
100   WRITE(6,101)                                                       ERR00190
101   FORMAT(5X,'INVALID TYPE FIELD')                                    ERR00200
      RETURN                                                             ERR00210
200   IF(IHELP.EQ.2) GO TO 22                                            ERR00220
      WRITE(6,201)                                                       ERR00230
201   FORMAT(5X,'QUERY INCOMPLETE')                                      ERR00240
      RETURN                                                             ERR00250
22    WRITE(6,202)                                                       ERR00260
202   FORMAT(5X,'QUERY INCOMPLETE OR NO SPACE DELIIMITER')               ERR00270
      RETURN                                                             ERR00280
300   WRITE(6,301)                                                       ERR00290
301   FORMAT(5X,'INVALID SELECT FIELD')                                  ERR00300
      RETURN                                                             ERR00310
500   GO TO (52,52,53,54,55,56,56,56,56,56,57,57,57,57,57,57,57,57,54,57ERR00320
     -),IHELP                                                           ERR00330
52    WRITE(6,501)                                                       ERR00340
501   FORMAT(5X,'MISSING WORD ''FOR'' IN CONDITION FIELD')               ERR00350
      RETURN                                                             ERR00360
53    WRITE(6,502)                                                       ERR00370
502   FORMAT(5X,'MISSING LEFT PARENTHESIS IN CONDITION FIELD')           ERR00380
      RETURN                                                             ERR00390
54    WRITE(6,503)                                                       ERR00400
503   FORMAT(5X,'MISSING SPACE DELIMITER IN CONDITION FIELD')            ERR00410
      RETURN                                                             ERR00420
55    WRITE(6,504)                                                       ERR00430
504   FORMAT(5X,'MISSING RIGHT PARENTHESIS IN CONDITION FIELD')          ERR00440
      RETURN                                                             ERR00450
```

```
56      WRITE(6,505)                                                    ERR00460
505     FORMAT(5X,'INVALID LOGICAL OPERATOR ''OR'',''AND'' ')           ERR00470
        RETURN                                                          ERR00480
57      WRITE(6,506)                                                    ERR00490
506     FORMAT(5X,'INVALID OPERAND FOR STATUS')                         ERR00500
        RETURN                                                          ERR00510
600     IF(IHELP.EQ.2) GO TO 54                                         ERR00520
        IF(IHELP.EQ.1) GO TO 61                                         ERR00530
        WRITE(6,601)                                                    ERR00540
601     FORMAT(5X,'INVALID RELATIONAL OPERATOR IN CONDITION FIELD')     ERR00550
        RETURN                                                          ERR00560
61      WRITE(6,602)                                                    ERR00570
602     FORMAT(5X,'INVALID FIRST OPERAND FOR RELOP IN CONDITION FIELD') ERR00580
        RETURN                                                          ERR00590
700     WRITE(6,701)                                                    ERR00600
701     FORMAT(5X,'INVALID PRINT FIELD')                                ERR00610
        RETURN                                                          ERR00620
999     IF(IER.NE.1001) GO TO 900                                       ERR00630
        WRITE(6,801)                                                    ERR00640
801     FORMAT(5X,'ERROR IN INPUT DATA - INVALID STATUS CODE')          ERR00650
        RETURN                                                          ERR00660
900     WRITE(6,901)                                                    ERR00670
901     FORMAT(5X,'ERROR IN INPUT DATA - DATA LESS THAN REQUIRED')      ERR00680
        RETURN                                                          ERR00690
        END                                                             ERR00700
C   ***************************************************************** TIM00010
C   *                                                               * TIM00020
C   *     SUBROUTINE TIMER PRINTS THE TIME IN SECONDS THE VIRTUAL   * TIM00030
C   *  MACHINE WAS USED SINCE THE LAST CALL.                        * TIM00040
C   ***************************************************************** TIM00050
C                                                                     TIM00060
        SUBROUTINE TIMER                                              TIM00070
        COMMON ITM(6),TIME                                            TIM00080
        CALL IXCLOK(ITM)                                              TIM00090
        SEC=ITM(5)/76800.                                             TIM00100
        EL=SEC-TIME                                                   TIM00110
        TIME=SEC                                                      TIM00120
        WRITE(6,100) EL                                               TIM00130
100     FORMAT(10X,'EXECUTION TIME FOR QUERY =',F8.4,'  SECONDS')     TIM00140
        RETURN                                                        TIM00150
        END                                                           TIM00160
```

```
C    ************************************************************** REW00010
C    *   PROGRAM REWRITE CONVERT THE STANDARD MPS OUTPUT INTO STANDARD  * REW00020
C    *   INPUT FORMAT FOR SYSTEM  ATHENA.                             * REW00030
C    *   FILE FT01F001 : OUTPUT OF MPS IN TAPES                       * REW00040
C    *   FILE FT02F001 : INTERMADIATE FILE IN DISK.                   * REW00050
C    *   FILE FT09F001 : INPUT FILE FOR ATHENA IN TAPE.               * REW00060
C    *                                                                * REW00070
C    ************************************************************** REW00080
// EXEC FORTCLG
//FORT.SYSIN DD *
      INTEGER NAME(2),A(20)                                           REW00090
      INTEGER*2 LIN(70),TCNF(6),STAT,MIN,BLANK,E                      REW00100
      DATA TCNF,BLANK,E/'0 ','.  ','1 ','E ','7 ','6 ','  ','E '/    REW00110
      DATA ISECT,INUM,IST,MIN,MPS/'SECT',' NUM','STAT','- ','MPS-'/   REW00120
      DEFINE FILE 2(16000,80,L,IA)                                    REW00130
      IFLGP=0                                                         REW00140
      IA=2                                                            REW00150
      MET=0                                                           REW00160
10    READ(1,100) LAB                                                 REW00170
100   FORMAT(1X,A4)                                                   REW00180
      IF(LAB.NE.ISECT) GO TO 10                                       REW00190
20    NROWS=MET                                                       REW00200
30    READ(1,200,ERR=30,END=99) LAB,(NAME(I),I=1,2),STAT,(LIN(J),J=1,70)REW00210
200   FORMAT(1X,A4,5X,2A4,2X,A2,5(2X,14A1))                           REW00220
      IF(LAB.EQ.ISECT) GO TO 20                                       REW00230
      IF(LAB.EQ.INUM.OR.LAB.EQ.MPS) GO TO 30                          REW00240
      IF(LAB.EQ.IST) GO TO 99                                         REW00250
      MET=MET+1                                                       REW00260
      DO 1 I=1,5                                                      REW00270
      IF(LIN(I*14).NE.MIN) GO TO 5                                    REW00280
      LIN(I*14)=BLANK                                                 REW00290
      K=(I-1)*14                                                      REW00300
      DO 2 J=1,14                                                     REW00310
      IF(LIN(K+J).EQ.BLANK) GO TO 2                                   REW00320
      LIN(K+J-1)=MIN                                                  REW00330
      GO TO 1                                                         REW00340
2     CONTINUE                                                        REW00350
5     IF(I.GT.4.OR.I.LT.3) GO TO 1                                    REW00360
      IF(LIN(I*14-1).NE.E) GO TO 1                                    REW00370
      K=I*14-6                                                        REW00380
      DO 3 J=1,6                                                      REW00390
      LIN(K+J)=TCNF(J)                                                REW00400
3     CONTINUE                                                        REW00410
      IF(I.EQ.4) GO TO 1                                              REW00420
      LIN(K)=MIN                                                      REW00430
1     CONTINUE                                                        REW00440
      WRITE(2'IA,500) (NAME(I),I=1,2),STAT,(LIN(J),J=1,70)            REW00450
500   FORMAT(2A4,A2,70A1)                                             REW00460
```

```
      GO TO 30                                                          REW00470
99    NCOL=MET-NROWS                                                    REW00480
      WRITE(2'1,600) NROWS,NCOL,IFLGP                                   REW00490
600   FORMAT(I5,30X,I5,10X,I1)                                          REW00500
      IA=1                                                              REW00510
      READ(2'IA,600) NROWS,NCOL,IFLGP                                   REW00520
      ITOT=NROWS+NCOL                                                   REW00530
      WRITE(9,600) NROWS,NCOL,IFLGP                                     REW00540
      DO 6 I=1,ITOT                                                     REW00550
      READ(2'IA,700) A                                                  REW00560
      WRITE(9,700) A                                                    REW00570
700   FORMAT(20A4)                                                      REW00580
6     CONTINUE                                                          REW00590
      END FILE 9                                                        REW00600
      STOP                                                              REW00610
      END                                                               REW00620
//GO.FT01F001 DD DISP=OLD,UNIT=3400-4,VOL=SER=(NPS251,NPS631),
// DSNAME=MPSLIST,DCB=(RECFM=UA,BLKSIZE=133,DEN=2)
//GO.FT02F001 DD DSN=S1951.STADFM,
// UNIT=3330,VOL=SER=DISK04,
// SPACE=(CYL,(6,1)),DISP=(NEW,KEEP),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//GO.FT09F001 DD UNIT=2400,VOL=SER=NPS713,DISP=(,KEEP),
// LABEL=(,SL),DSN=TAPFOR,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400,DEN=2)
```

LIST OF REFERENCES

1. Aho, A.V. and Ullman, J.D., Principles of Compiler Design, Addison-Wesley, 1977.

2. Bradley, G.H., Brown, G.G. and Graves, G.W., Preprocessinng of Large Scale Linear Programs (in preparation), March 1979.

3. Brown, G. and Graves, G., XS - An Experimental System for LARGE SCALE MIXED INTEGER OPTIMIZATION with Elastic Programming, National ORSA/TIMS convention, Las Vegas, Navada, Nov. 17, 1975.

4. International Business Machines Inc., Mathematical Programming System/360 Version 2, Linear and Separable Programming - User's Manual, 1971.

5. International Business Machines Inc., Control Program 67/Cambridge Monitor System (CP/CMS), Version 3.25, 1974.

6. Mavrikas, C., Optimal 5-year Planning Using Mixed-Integer Linear Programming. Three Models Implemented for Naval Air Test Center, M.S. Thesis, Naval Postgraduate School, Monterey, 1979.

7. O;Neil, R.P. and Sanders, R.C., PERUSE System Manual Version 2, Sept. 1977.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	2
4.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	2
5.	Professor Gordon Bradley, Code 52BZ Naval Postgraduate School Monterey, California 93940	2
6.	Professor Gerald Brown, Code 55BW Naval Postgraduate School Monterey, California 93940	1
7.	Greek Army Headquarters Department of Research and Design Holargos, Athens, GREECE	2
8.	Captain Panagiotis I. Galatas Alketou 19, Pagrati Athens 506, GREECE	3
9.	Professor Glenn W. Graves Graduate School of Management University of California Los Angeles, California 90024	1